UNIVERSIDAD POLITÉCNICA DE MADRID
ESCUELA TÉCNICA DE TELECOMUNICACIONES

PROYECTO FIN DE CARRERA

TRANSFORMACIÓN DEL ESPACIO CINEMÁTICO LABAN PARA LA EVALUACIÓN DEL MOVIMIENTO

Mª ILENIA DIZ MIGUEL

CURSO 2013-2014

E.T.S.I.S. TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA
PLAN 2000

TEMA: Estudio del movimiento corporal con ayuda de una red neuronal.

TÍTULO: Transformación del espacio cinemático Laban para la evaluación del movimiento.

AUTOR: Mª Ilenia Diz Miguel

TUTOR: Lino García Morales **Vº Bº.**

DEPARTAMENTO: DIAC

Miembros del Tribunal Calificador:

PRESIDENTE: Juana Sendra Pons

VOCAL: Lino García Morales

VOCAL SECRETARIO: Antonio Mínguez Olivares

DIRECTOR:

Fecha de lectura: 17 de Septiembre de 2014

Calificación: **El Secretario,**

RESUMEN DEL PROYECTO:

Evaluación del movimiento de una persona a través de las acciones de esfuerzo de Laban. Gracias al análisis del movimiento que realizó Laban se puede caracterizar como es el movimiento de una persona, aunque es una técnica que se utiliza sobre todo en danza y arte dramático, en este caso se va a aplicar para analizar el movimiento en poblaciones con riesgo de exclusión como es el espectro autista.
Con las acciones de esfuerzo no sólo se caracteriza el movimiento dinámico del cuerpo sino que también se podría llegar a relacionar la manera de moverse de la persona con su estado emocional. El movimiento del cuerpo en el espacio depende de la combinación de los cuatro factores de acciones de esfuerzo: peso, tiempo, espacio y flujo, y estos a su vez se clasifican en dos elementos opuestos, elementos de acción de esfuerzo.

El objetivo es crear y entrenar un sistema que a partir de unos puntos cinemáticos (persona en movimiento) a su entrada, obtenga la clasificación del movimiento en uno de estas 8 acciones de esfuerzo, a su salida. Para ello se cuenta con el esqueleto que ofrece kinect que consta de 15 articulaciones (pies, rodillas, caderas, hombros, codos, manos, torso, cuello y cabeza), y gracias a un proyecto previo se obtiene la posición, velocidad y aceleración de dichas articulaciones.

AGRADECIMIENTOS

A mi madre y a mi hermana, por estar siempre ahí.

A Alberto, por tu paciencia y apoyo.

A mi tutor, por su ayuda.

No sobrevive el más fuerte sino el que mejor se adapta.
Charles Darwin

Como fuerza social, un individuo con una idea vale
por noventa y nueve con un solo interés.
John Stuart Mill

Los que sueñan de día son conscientes de muchas cosas
que escapan a los que sólo sueñan de noche.
Edgard Allan Poe

Resumen

Este proyecto tiene como objetivo la implementación de un sistema capaz de analizar el movimiento corporal a partir de unos puntos cinemáticos. Estos puntos cinemáticos se obtienen con un programa previo y se captan con la cámara kinect.

Para ello el primer paso es realizar un estudio sobre las técnicas y conocimientos existentes relacionados con el movimiento de las personas. Se sabe que Rudolph Laban fue uno de sus mayores exponentes y gracias a sus observaciones se establece una relación entre la personalidad, el estado anímico y la forma de moverse de un individuo. Laban acuñó el término esfuerzo, que hace referencia al modo en que se administra la energía que genera el movimiento y de qué manera se modula en las secuencias, es una manera de describir la intención de las expresiones internas. El esfuerzo se divide en 4 categorías: peso, espacio, tiempo y flujo, y cada una de estas categorías tiene una polaridad denominada elemento de esfuerzo. Con estos 8 elementos de esfuerzo un movimiento queda caracterizado. Para poder cuantificar los citados elementos de esfuerzo se buscan movimientos que representen a alguno de ellos.

Los movimientos se graban con la cámara kinect y se guardan sus valores en un archivo *csv*. Para el procesado de estos datos se establece que el sistema más adecuado es una red neuronal debido a su flexibilidad y capacidad a la hora de procesar entradas no lineales. Para la implementación de la misma se requiere un amplio estudio que incluye: topologías, funciones de activación, tipos de aprendizaje, algoritmos de entrenamiento entre otros. Se decide que la red tenga dos capas ocultas, para mejor procesado de los datos, que sea estática, siga un proceso de cálculo hacia delante (*Feedforward*) y el algoritmo por el que se rija su aprendizaje sea el de retropropagación (*Backpropagation*) En una red estática las entradas han de ser valores fijos, es decir, no pueden variar en el tiempo por lo que habrá que implementar un programa intermedio que haga una media aritmética de los valores.

Una segunda prueba con la misma red trata de comprobar si sería capaz de reconocer movimientos que estuvieran caracterizados por más de un elemento de esfuerzo. Para ello se vuelven a grabar los movimientos, esta vez en parejas de dos, y el resto del proceso es igual.

Abstract

The aim of this project is the implementation of a system able to analyze body movement from cinematic data. This cinematic data was obtained with a previous program.

The first step is carrying out a study about the techniques and knowledge existing nowadays related to people movement. It is known that Rudolf Laban was one the greatest exponents of this field and thanks to his observations a relation between personality, mood and the way the person moves was made. Laban coined the term effort, that refers to the way energy generated from a movement is managed and how it is modulated in the sequence, this is a method of describing the inner intention of the person. The effort is divided into 4 categories: weight, space, time and flow, and each of these categories have 2 polarities named elements of effort. These 8 elements typify a movement. We look for movements that are made of these elements so we can quantify them.

The movements are recorded with the kinect camera and saved in a *csv* file. In order to process this data a neural network is chosen owe to its flexibility and capability of processing non-linear inputs. For its implementation it is required a wide study regarding: topology, activation functions, different types of learning methods and training algorithms among others. The neural network for this project will have 2 hidden layers, it will be static and follow a feedforward process ruled by backpropagation. In a static net the inputs must be fixed, this means they cannot vary in time, so we will have to implement an intermediate program to calculate the average of our data.

A second test for our net will be checking its ability to recognize more than one effort element in just one movement. In order to do this all the movements are recorded again but this time in pairs, the rest of the process remains the same.

Contenido

Índice de figuras

Índice de tablas

Índice de códigos

Capítulo 1

Introducción

1.1 Introducción general, motivación y objetivos

El movimiento corporal ha sido objeto de estudio por muchas personas dedicadas al mundo de la danza y del arte dramático, pero fue a partir de las investigaciones de Rudolph Laban cuando comenzó a asociarse con la conducta del individuo. Este definió una serie de acciones que describían los movimientos de los bailarines. Este proceso siempre se llevaba a cabo de una manera subjetiva dejando recaer toda la responsabilidad sobre el conocimiento del observador.

La finalidad de este proyecto es la de implementar un sistema capaz de reconocer esas categorías de movimientos. Para ello se necesita crear y entrenar una red neuronal que procese datos cinemáticos a su entrada, extraídos del movimiento del cuerpo de una persona, y obtenga su clasificación en dichas categorías.

El avance de la tecnología ha permitido al ser humano salvar muchas barreras. Son claros los ejemplos que incumben nuestras tareas cotidianas pero de mayor importancia son aquellos que logran mermar una minusvalía o una deficiencia. Este proyecto busca contribuir en este campo buscando una manera de interactuar con personas dentro del espectro autista a través del análisis del movimiento corporal. Se pretende establecer las bases y abrir unas vías hacia un trabajo más ambicioso.

1.1.1 Herramientas de trabajo

Para llevar a cabo el desarrollo del programa se va a utilizar el entorno de programación Processing 2.0.3 basado en Java que además de ser libre goza de un alto nivel de compatibilidad entre plataformas, la librería de redes neuronales Encog 3.1.0 para Java y la librería SimpleOpenNI para interactuar con kinect. En el anexo A se puede ver cómo efectuar su instalación.

1.1.2 Estado del arte

La mayoría de los avances realizados en el estudio de movimientos se han llevado a cabo en el ámbito académico. Se parte de sistemas que recogen datos a través de cámaras de vídeo y sensores hasta llegar a los acelerómetros que se utilizan hoy en día. Algunos los relacionan con el estudio realizado por Laban y otros simplemente se centran en el reconocimiento de diferentes patrones.

Liwei Zao en su tesis utilizaba un programa denominado EMOTE (Expressive MOTion Engine) con el que recrea animaciones de movimientos con combinaciones de *esfuerzo* y *forma*. Zao cuenta con expertos en LMA (*Laban Movement Anlysis*) que realizan los movimientos acordados y consiguen relacionar variables como grados de curvatura, velocidad, aceleración, recorrido con las diferentes subcategorías del *esfuerzo*. Para procesar estos datos cuentan con 4 redes neuronales, una por cada subcategoría de *esfuerzo* (Zao, 2001).

Otro proyecto que relaciona variables de movimiento con el estudio realizado por Laban, fue llevado a cabo por (Jui-Fa, Wei-Chuan, Kun-Hsiao, & Shih-Yao, 2011). En este caso contaban con un sistema bastante complejo de recogida de datos, 45 sensores que colocaban sobre el cuerpo del bailarín, 2 videocámaras, una estación de almacenaje de datos y dos ordenadores entre otros. En este caso también contaban con expertos que juzgaban los movimientos del bailarín mientras los relacionaban con variables de velocidad y aceleración.

Los inconvenientes de estos sistemas basados en Visión Artificial (*Computer Vision*), son que en muchos casos dependen de la iluminación, del ángulo que con el que se capte la imagen y que requieren de diversos elementos de *hardware*.

La llegada de kinect al mercado simplificó notablemente el proceso de recogida de datos, ya que con un solo instrumento se cuenta con una imagen RGB además de la de profundidad que es la que da la información de distancia. Un ejemplo de su utilización (Hall J. C., 2011), destacar que en este caso no se utiliza una red neuronal para el tratamiento de los datos sino un Modelo Oculto de Márkov (*Hidden Márkov Model*, HMM), un proceso estadístico.

Actualmente está emergiendo una técnica llevada a cabo con acelerómetros. La gran ventaja de estos es que se encuentran en muchos dispositivos de uso común como puede ser un *Smartphone* (Jiahui, Gang, Daqing, Guande, & Shijian, 2013). Se obtienen muy buenos resultados de la combinación de acelerómetros con el uso de HMM ó FDSVM (*Frame-based Descriptor and multi-class Support Vector Machine*).

Capítulo 2

Estudio del gesto

2.1 Gesto

Los gestos son movimiento (del rostro, de las manos, del cuerpo, etc.) con que se expresan afectos del ánimo. Es decir, movimientos cargados de significado según el contexto (cultura, situación, etc.). Los gestos se pueden entender, por lo tanto, como un conjunto de señales dinámicas que dan a entender algo, que exigen una interpretación, como un lenguaje.

2.1.1 ¿Qué es la expresión corporal?

La comunicación no verbal se define como la comunicación mediante expresión o lenguaje corporal desprovisto de palabras. Estos mensajes pueden ser comunicados a través de gestos, lenguaje corporal, la postura, expresión facial, contacto visual, etc.

La comunicación no verbal es anterior a la evolución del lenguaje pero debido a su compleja interpretación apenas hay estudios realizados anteriormente a 1950. En el artículo: *The expression of emotions in animals and man* (1873), Charles Darwin, estudia la forma de expresión de las emociones y llega a la conclusión de que los patrones de respuesta expresiva emocional son innatos y que existen programas genéticos que determinan la forma de la respuesta de la expresión emocional siendo los modos más importante de respuesta a través de reflejos e instinto (Chóliz, 1995).

A partir de 1950 se desarrollan estudios de todas las áreas relacionadas con la comunicación humana, incluida la comunicación no verbal donde cabe destacar los siguientes trabajos:

- Ray Birdwhistell, antropólogo, fue el primero en utilizar la palabra *cinestesia* o *kinesthesia* o *quinestesia*: rama de la ciencia que estudia el movimiento humano e interpretación del comportamiento no verbal o lenguaje corporal. Birdwhistell defendía la idea de que los movimientos del cuerpo durante una conversación con otra persona podían ser analizados de la misma manera que el lenguaje hablado (Birdwhistell, 1952).

- Jürgen Ruesch y Weldon Kees, psiquiatra y fotógrafo, respectivamente, publicaron el primer libro que utilizó el término "no verbal" (Ruesch & Kees, 1972). Con su trabajo proporcionan conocimiento teórico adicional, del origen de la codificación y de la utilización de la conducta no-verbal además de abundante documentación gráfica.

- Edward T. Hall, antropólogo, publicó (Hall E. T., 1959), en el que explora el contexto multicultural de la comunicación. Defiende la idea de que la mayoría de la comunicación es no-verbal y siempre sigue patrones culturales. Acuñó conceptos como "acento espacial" analizando comportamientos culturales específicos como la zona invisible asociada a cada individuo.

- Otros investigadores han realizado importantes contribuciones, como Flora Davis, Allan Pease, Desmond Morris y Paul Ekman.

Por tanto, la expresión corporal queda englobada dentro de la comunicación no verbal y se define a su vez como (Castañer, 2000):

> "*Toda manifestación de movimiento de nuestro cuerpo sea estática (postura) o dinámica (gesto) se puede considerar como forma de expresión siendo capaz de transmitir a través de los recursos corporales, de forma consciente o no, una información que suele ser referida a sensaciones y emociones*."

La expresión corporal es un medio de comunicación con el entorno, con otros sujetos y con uno mismo. Aunque mayoritariamente se utiliza en el mundo del arte, es esencial en el desarrollo de la formación del sujeto (Blanco Vega, 2009).

La investigación de la expresión corporal proviene de cinco disciplinas diferentes: la psicología, la psiquiatría, la antropología, la sociología y la etología. Los psiquiatras reconocen desde hace mucho tiempo que la forma de moverse de un individuo proporciona datos ciertos sobre su carácter, sus emociones y las reacciones hacia la gente que lo rodea.

2.1.2 Expresión de emociones a través de movimientos

Los estudios de la relación del movimiento y el estado emocional del individuo llevaron al desarrollo de la *Danzaterapia*. Esta disciplina impulsada sobre todo durante los años 40s y 50s, combinaba movimientos de la danza moderna con terapias psicológicas y psiquiátricas. Su máxima era que el movimiento corporal refleja el estado de las emociones internas y que los cambios en los movimientos llevan a cambios en la mente (Levy, 1988).

Hubo grandes investigadores de esta materia pero cabe destacar a dos de ellos por encima del resto, Rudolph Laban, maestro de danza moderna húngaro creador de la *labanotación*, y su estudiante Imgard Bartenieff.

El *Análisis de Movimiento* de Laban (LMA, *Laban Movement Analysis*), es un sistema que permite registrar el movimiento corporal, deriva de la notación de la danza (en cuanto documenta todas las poses del movimiento humano) y tiene como finalidad conocer las intenciones de la persona examinada así como actitudes internas que preceden a la acción.

El modelo de Laban se divide en 4 categorías: forma, cuerpo, esfuerzo y espacio. Estas 4 categorías que a su vez se subdividen como se explica a continuación, permiten la

descripción de todos los aspectos del movimiento, tanto consciente como inconsciente (Konie, 2011). Véase Figura 1.

- **Cuerpo:** Esta categoría es la unión de la conectividad interior y la expresión exterior. Describe el movimiento físico del cuerpo así como de sus miembros. Esta categoría fue desarrollada por Imgard Bartenieff, alumna de Laban. Se divide en: Iniciación del movimiento, conexión entre las diferentes partes del cuerpo, secuencia del movimiento y patrones neuromusculares.

- **Esfuerzo:** Es una característica tanto funcional como expresiva, con él analizamos las cualidades más sutiles del movimiento que ayudan a entender la intención del movimiento. El esfuerzo se divide en 4 factores: Espacio, peso, tiempo y flujo, estos factores tienen dos polaridades respectivamente: Directo/Indirecto, Pesado/Liviano, Súbito/Sostenido y Guiado/Libre. La combinación de los 4 factores y sus polaridades dan lugar a las **Acciones de esfuerzo**, que a continuación se estudian con más detalle.

- **Forma:** Hace referencia a la forma en la que el cuerpo cambia durante el movimiento. Es la unión entre cuerpo y espacio. Esta categoría se divide en: Formas estáticas, modos de cambio de la figura, cualidades de la forma y apoyo al flujo.

- **Espacio:** Describe la relación del movimiento con el entorno. Se subdivide en: Kinesfera, intención espacial y geometría.

Con la combinación de las categorías *esfuerzo-forma* se logran deducir hechos relacionados con el carácter del hombre no por la realización de un movimiento particular sino por el estilo integral en que se mueve. El analista de *esfuerzo-forma* estudia el flujo del movimiento, en tensión o en relajación, intenso o leve, repentino o directo, etc. y en la *forma*, que es en realidad un concepto de la danza: las formas que adopta el cuerpo en el espacio (Davis, 2010).

Este proyecto se centra en la obtención e identificación de cada una de las diferentes subcategorías de esfuerzo como descriptor del gesto. El esfuerzo proporciona información acerca de la cualidad del movimiento y como se ha dicho anteriormente el movimiento es una forma de comunicación no verbal a través de su estudio puede llegar a saberse el estado emocional de una persona así como rasgos de su personalidad, ambas características muy útiles en cuanto a su aplicación en rangos de población con minusvalías o dificultad para expresarse.

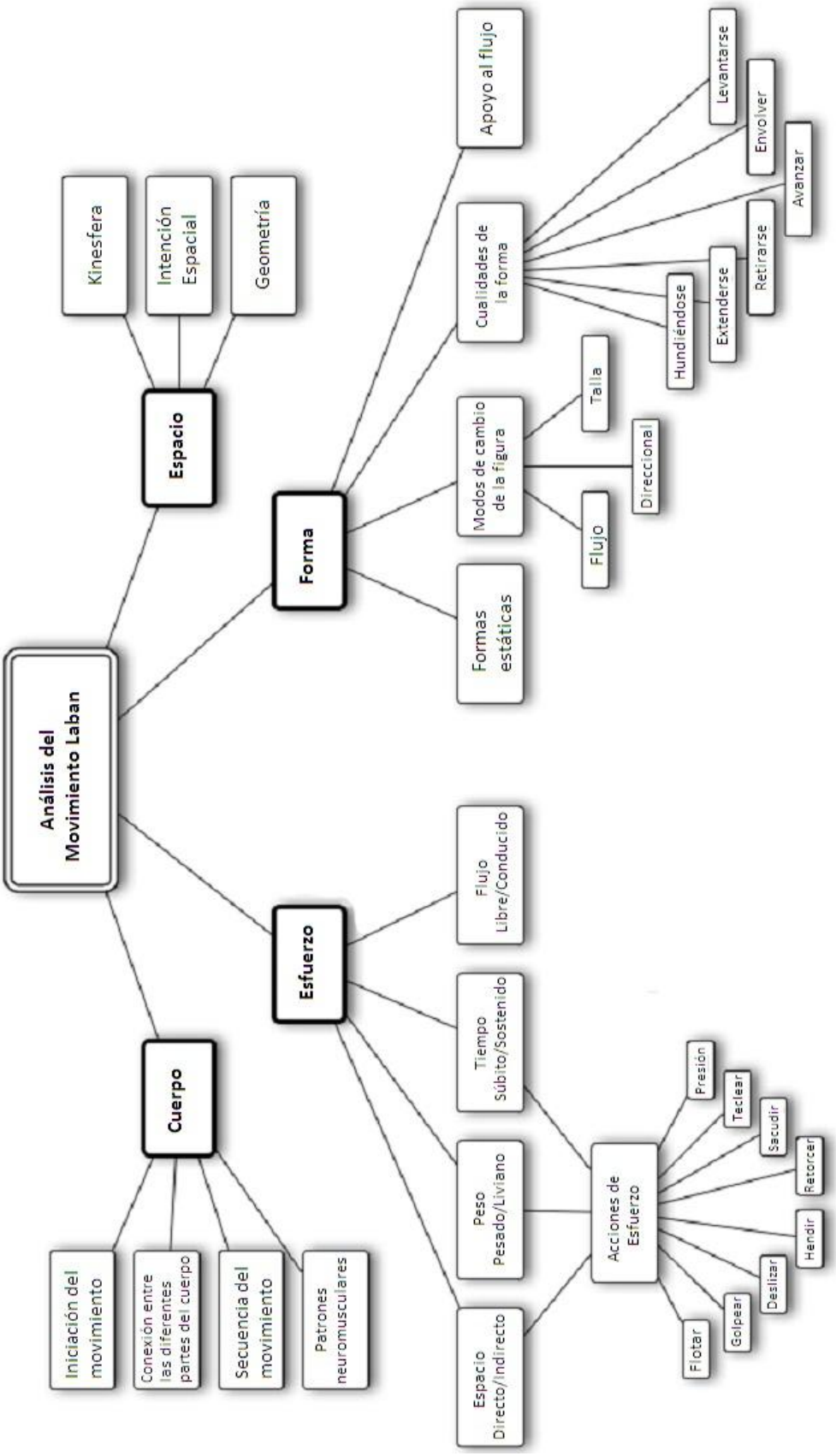


Figura 1: Esquema de la división de los componentes del Análisis de Movimiento de Laban.[1]

[1] Adaptación de http://blog.lib.umn.edu/ali/2009arts3305/2009/01/laban_movement_analysis_1.html

2.2 Modelo cinemático

Este proyecto tiene su punto de partida en un modelo cinemático obtenido previamente (Ramos, 2013). Este modelo cinemático consta de un esqueleto formado por 15 articulaciones: pies, rodillas, caderas, hombros, codos, manos, torso, cuello y cabeza, de los cuales obtenemos la aceleración y velocidad instantáneas por cada trama.

El registro del esqueleto se realiza con la cámara Kinect de PrimeSense, que toma una imagen RGB de la escena y otra de profundidad. Este último tipo de imágenes son mucho más útiles ya que no dependen de las condiciones de luz con las que se registra. Cada píxel toma un color dependiendo de la distancia a la que se encuentran siendo más fácil el procesado por un ordenador porque es capaz de distinguir de esta manera donde empieza y acaba un objeto, persona, etc.

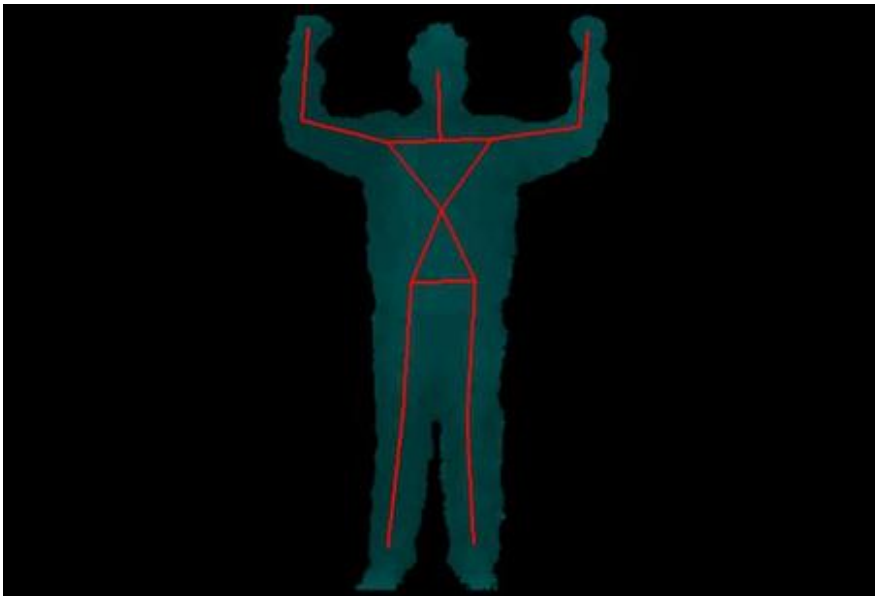


Figura 2: Esqueleto detectado por kinect[2]

2.3 Análisis de movimiento

La danzaterapia es el uso psicoterapéutico del movimiento, en el cual se ponen de manifiesto aspectos emocionales, cognitivos, físicos y sociales de cada individuo.[3]

Las primeras aproximaciones a la danzaterapia surgieron en el hospital psiquiátrico Jacobi, donde se puso de manifiesto el aspecto terapéutico de la danza, mediante la cual los pacientes eran capaces de expresar sentimientos, relacionarse ó modificar su actitud. Mediante el estudio y modificación de la manera de moverse de los pacientes se conseguía un cambio en su conducta. La danzaterapia ofrece un medio de comunicación no verbal para personas con dificultades en el habla o capacidad verbal subdesarrollada. Se encuentra especialmente útil en el tratamiento del espectro autista.

[2] http://www.techshout.com/software/2010/31/faast-toolkit-for-full-body-control-over-games-works-with-kinect/
[3] American Dance Therapy Association

Imgard Bartenieff fue una de las fundadoras de la Asociación Americana de Danzaterapia, en la que combina sus conocimientos en rehabilitación física con LMA (Connett, 2011). Nació en Berlín en 1900 y durante su juventud estudió danza. En 1925 comenzó sus estudios con Laban, durante dos años aprendió sobre su clasificación de movimientos (especialmente sobre las componentes de esfuerzo y espacio), al que ayudó en el desarrollo de su sistema de notación. En 1936 emigró a EE.UU. Es justo aquí donde comienza una nueva manera de integrar todo lo aprendido sobre movimiento en danza a la rehabilitación, tanto física como mental. En 1943 se graduó en Terapia Física por la Universidad de Nueva York y trabajó hasta 1953 con enfermos de polio aplicando sus técnicas. De 1957 a 1967, trabajó como danzaterapeuta e investigadora del comportamiento no verbal en el hospital Albert Einstein. Tomando el trabajo de Laban como base desarrolló un sistema de observación y notación del comportamiento del paciente. Un ejemplo de cómo la observación del movimiento en pacientes se podía relacionar con sus afecciones se describe en el libro *La comunicación no verbal* de Flora Davis, en el cual durante una sesión de terapia Irmgard y Martha Davis, asistente del hospital, observan a una joven. Martha describió el comportamiento como enérgico y Bartenieff llegó a la conclusión de que estaba deprimida y tenía intenciones suicidas. Bartenieff justificó este diagnóstico en que la paciente iniciaba movimientos muy rápidos, algo poco común, que se transformaban en movimientos forzados y directos, como si ella misma cortara ese movimiento, no lo dejara terminar un patrón poco frecuente. El éxito de la danzaterapia reside en la creencia de que el cuerpo y mente están conectados y haciendo un cambio en alguno de ellos, repercute en el otro.

Rudolf Laban (1879-1958), fue maestro de danza moderna y su trabajo de análisis de movimiento, conocido como LMA, sentó las bases para un nuevo sistema de evaluación muy utilizado hoy en día por atletas, bailarines, terapeutas, etc. (Trinity Laban Conservatoire of Music and Dance) Aunque originariamente era pintor y arquitecto se dedicó al mundo de la danza. En sus primeros años de carrera escribió numerosos artículos y libros referentes a la actuación y al movimiento e inventó un sistema para registrar todos los movimientos de los bailarines denominado *Labanotación* (Labanotation). Durante la década de 1920 llegó a abrir 25 escuelas por diferentes países de Europa donde enseñaba desde niños a bailarines profesionales, estas escuelas le sirvieron para continuar con su investigación del movimiento. En el año 1938, con el auge del nazismo en Europa, se vio forzado a emigrar a Gran Bretaña, donde cambió el rumbo de sus investigaciones, estudiando la eficiencia y la fatiga en la industria. Laban intentó encontrar secuencias de movimiento que fueran variadas y evitaran cualquier tipo de esfuerzo innecesario al obrero, gracias a estos hallazgos desarrolló un sistema de análisis de movimiento diferente, a través del esfuerzo (*Effort*). En sus últimos años de vida centró sus estudios en el movimiento como comportamiento, estudiando las necesidades conductuales de trabajadores de la industria y de pacientes psiquiátricos.

2.3.1 Esfuerzo

El esfuerzo hace referencia al modo en que se administra la energía que genera el movimiento y de qué manera se modula en las secuencias (Newlove & Dalby, 2004). Es una manera de describir la expresión de las intenciones internas. Con un pequeño cambio en la organización del cuerpo, un movimiento realizado con diferente esfuerzo puede tener un efecto muy diferente.

El esfuerzo tiene 4 subcategorías, también denominadas factores de esfuerzo, y cada una de ellas con diferentes polaridades, también conocidas como elementos de esfuerzo: *peso* (pesado/liviano), *espacio* (directo/indirecto), *tiempo* (sostenido/súbito) y *flujo* (conducido/libre). Estas categorías se pueden representar mediante un gráfico.

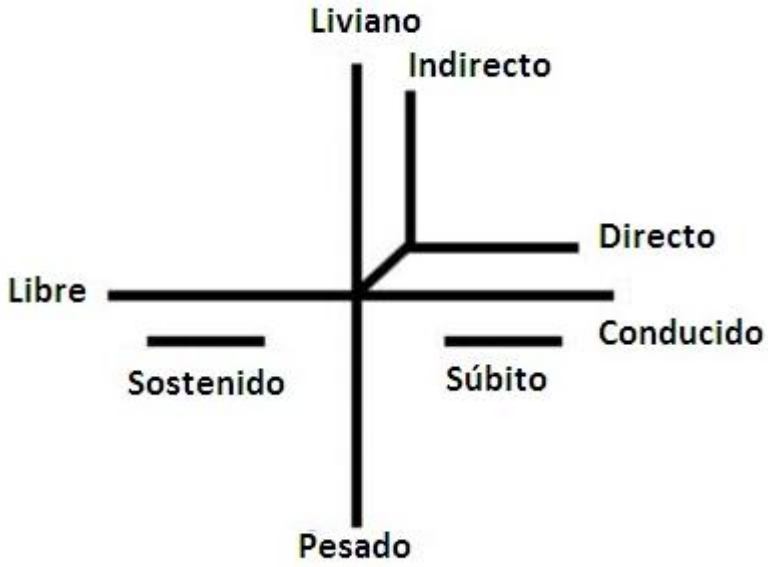


Figura 3: Gráfico de elementos de esfuerzo.[4]

2.3.1.1 Peso

El *peso*, por definición, es la fuerza ejercida por un campo gravitacional. El peso de un cuerpo es la fuerza con que la Tierra lo atrae y se mide en kilogramos. Pero el peso relacionado con los movimientos poco tiene que ver con esta definición.

El peso, como movimiento, se caracteriza principalmente por cómo cada persona se deja llevar por la gravedad ó se resiste a ella. Un claro ejemplo de esto sería una persona obesa ejecutando movimientos con la ligereza de una pluma. El cuerpo siente cuanta fuerza hay que hacer para realizar un movimiento, a esta fuerza se le denomina fuerza cinética. Incluso cuando parece que el cuerpo está en una posición estática puede estar realizando una fuerza para mantener dicha posición. También está la fuerza que tenemos que usar frente a obstáculos, no es lo mismo empujar la sillita de un niño que intentar mover un coche, aquí en ambos casos la fuerza se aplica sobre una resistencia externa. Otro caso que pone a prueba nuestro sentido cinético es al levantar peso, no es lo mismo levantar una hoja de papel que un cubo de agua. En relación al peso los movimientos pueden ser pesados ó livianos y según la tensión, suaves ó fuertes.

2.3.1.2 Espacio

Para poder entender el espacio que habitamos necesitamos límites, señales, alguna referencia con la que poder decir si es grande o pequeño. Con tres coordenadas

[4] Adaptación de http://www.directornotation.org/acting7.html

seríamos capaces de situarnos en él, por ejemplo, la distancia a dos paredes de la habitación dónde nos encontramos y la altura. Necesitamos espacio para movernos y cuando lo hacemos nuestro cuerpo también desplaza espacio. El movimiento existe con nosotros.

El espacio que nos rodea se denomina *kinesfera*. El uso de este espacio personal se puede maximizar ó minimizar, ya sea bien doblándonos todo lo posible o estirándonos, estos movimientos fueron denominados por Laban como *recogerse* (*gathering*) y *dispersarse* (*scattering*). El grado de extensión es fundamental por las implicaciones emocionales y expresivas que sustenta. No sólo el cuerpo como un todo tiene esta propiedad sino sus partes por separado. Los brazos son un ejemplo, podemos abrir nuestros brazos, dispersarlos para saludar a un amigo y podemos recogerlos en un abrazo. Todos tenemos una manera de movernos y de utilizar el espacio que nos rodea acorde a nuestra personalidad, nuestros genes ó la situación personal por la que atravesemos, estos son sólo algunos parámetros que definen nuestro movimiento .

El espacio se divide en *directo* e *indirecto*. Es directo cuando corres en una carrera e indirecto cuando deambulas por una calle. Esta división depende de la cantidad de espacio que utilicemos para movernos y la decisión con que lo hagamos.

2.3.1.3 Tiempo

El tiempo y el espacio están unidos inextricablemente. Así como el tiempo y el ritmo. Se tiende a pensar en el ritmo como algo audible pero también puede ser algo visual.

Nuestros movimientos en el día a día se rigen por un ritmo libre y arrítmico. El corazón por ejemplo tiene un ritmo de latido dependiendo de si está en estado de reposo o en excitación, así como la respiración puede tener un ritmo constante en estado de calma.

El tiempo puede dividirse en *sostenido* ó *súbito*, se diferencian en la *duración*, un movimiento súbito es aquel que es corto en el tiempo y un movimiento sostenido lo contrario. Los movimientos *sostenidos* son más difíciles de determinar (bostezar, por ejemplo), algunos movimientos empiezan despacio y terminan deprisa como golpear una pelota con un palo de golf ó al revés, como sería el de un corredor entrando por la línea de meta, que no se para en seco sino que prosigue gradualmente bajando el ritmo hasta detener su marcha. Son movimientos con una duración determinada. El movimiento súbito es un movimiento impulsivo (como parpadear, sacudirse algo del abrigo, etc.).

2.3.1.4 Flujo

Se asocia el flujo al movimiento de un líquido, contra más ralo sea más fluido. El flujo describe un movimiento continuo y está relacionado con el grado de libertad de este. El flujo es *conducido* cuando se puede interrumpir en cualquier punto de su recorrido y es *libre* cuando no se puede detener en cualquier punto.

El movimiento fluye cuando la energía que generase propaga a través de diferentes articulaciones. Según el tipo de movimiento pueden transmitirse varias energías a la vez en direcciones contrarias ó distintas. Pintar una pared con una brocha es un ejemplo de flujo libre; se produce un movimiento del brazo completo con cierta sensación de libertad, sin embargo, si estuviéramos pintando el marco de una ventana en el que tenemos la mano firme y vamos con cuidado de no pintar el cristal sería un flujo conducido y no libre.

2.3.2 Las 8 acciones básicas del esfuerzo

Conocidas las calidades del movimiento se definen las acciones de esfuerzo: presionar, golpear, retorcer, dar latigazos, deslizar, teclear, flotar y sacudir (Schinca, 2002). Estas acciones están compuestas por la variación del tiempo, el peso, la dirección del movimiento y la fuerza con que se ejecutan. Las acciones de esfuerzo también pueden ser representadas mediante gráficos.

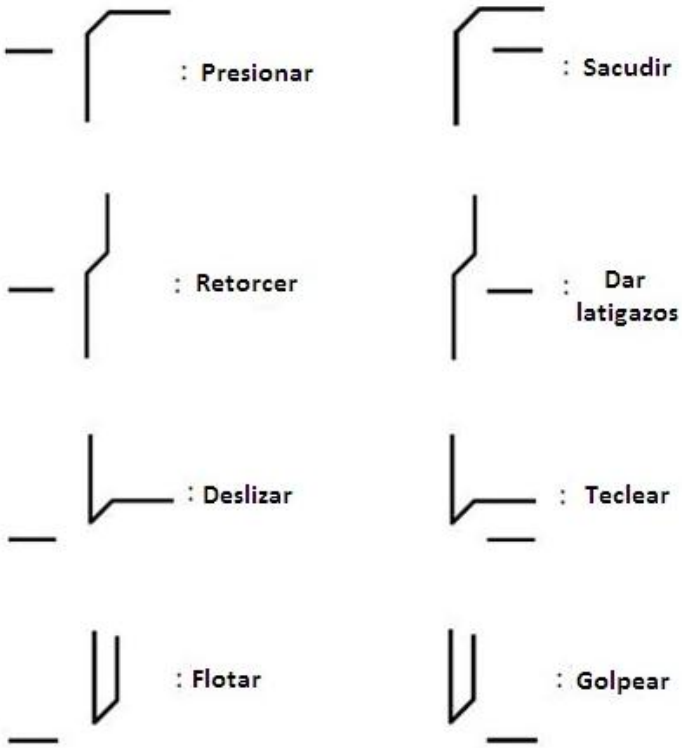


Figura 4: Gráfico de acciones de esfuerzo.[5]

2.3.2.1 Golpear

Este movimiento es de carácter violento. Un ejemplo claro es golpear con los puños cerrados un objetivo. Es directo, súbito, pesado y su flujo puede ser libre ó conducido.

[5] Adaptación de http://www.directornotation.org/acting6.html

2.3.2.2 Presionar

En nuestra vida diaria hay muchos movimientos que requieren que se ejerza presión. Algunos necesitan una presión más ligera como pueden ser, llamar al timbre ó clavar una chincheta y también los hay que requieren de más presión como planchar ó empujar un carro de la compra. Presionar es un *esfuerzo* muy pesado, muy directo y muy sostenido, además necesita de muy poco espacio para ejecutarlo. En cuanto al flujo es de carácter conducido.

2.3.2.3 Retorcer

Es el movimiento que se efectúa cuando escurrimos una bayeta mojada. El *esfuerzo* de retorcer implica no sólo las manos sino todo el cuerpo. El espacio es indirecto, el tiempo sostenido y el peso pesado, el flujo es conducido.

2.3.2.4 Dar latigazos

Al contrario que en los casos anteriores hendir no es un movimiento que realicemos a menudo. Este movimiento se realiza blandiendo una espada en combate ó una raqueta jugando al tenis. Hendir como *esfuerzo* es súbito, pesado, indirecto y libre.

2.3.2.5 Sacudir

Esta acción es muy común, es llevada a cabo cuando nos sacudimos la ropa, apartamos bichos de la comida ó nos apartamos el pelo de la cara. Es un *esfuerzo* de espacio indirecto, tiempo súbito y peso liviano, el flujo es libre.

2.3.2.6 Flotar

Este movimiento es parecido a volar, bien puede realizarse en el aire o en agua. También cuando caminamos con los brazos extendidos y de una manera muy ligera puede dar la sensación de flotar. Como *esfuerzo* es indirecto, sostenido y liviano, su flujo puede ser libre ó conducido.

Muy pocas tareas cotidianas pueden ser realizadas con un solo *esfuerzo*, para completarlas nos llevará

2.3.2.7 Deslizar

Es el movimiento que asociamos al patinaje sobre hielo, algo que se desarrolla con los pies aunque también puede darse con otras partes del cuerpo como por ejemplo al sentir una prenda de sea, deslizando nuestras manos sobre ella. Como *esfuerzo* deslizarse es sostenido, liviano, directo y conducido.

2.3.2.8 Teclear (dar golpecitos)

Otro tipo de movimiento muy común, tecleamos casi a diario en nuestro teclado del ordenador, pero también se puede considerar el mismo tipo de movimiento a dar toquecitos con el pincel sobre un lienzo. Este movimiento puede realizarse no sólo con las manos sino con otras partes del cuerpo. Es un *esfuerzo* de espacio directo, tiempo súbito, y peso liviano, generalmente el flujo es libre aunque también puede ser conducido.

La Tabla 1 muestra un resumen de la relación de las ocho acciones básicas del esfuerzo con los cinco esfuerzos. Observe que tales relaciones están declaradas con términos no objetivos, sino subjetivos.

	Espacio	**Tiempo**	**Peso**	**Flujo**	**Tensión**
Golpear	Directo	Súbito	Pesado	Libre ó conducido	Fuerte
Presionar	Directo	Sostenido	Pesado	Conducido	Fuerte
Retorcer	Indirecto	Sostenido	Pesado	Conducido	Fuerte
Hendir	Indirecto	Súbito	Pesado	Libre	Fuerte
Sacudir	Indirecto	Súbito	Liviano	Libre	Suave
Flotar	Indirecto	Sostenido	Liviano	Libre ó conducido	Suave
Deslizar	Directo	Sostenido	Liviano	Conducido	Suave
Teclear	Directo	Súbito	Liviano	Libre ó conducido	Suave

Tabla 1: Resumen de los distintos *esfuerzos*.

Como se ha explicado durante todo este capítulo, a través del estudio del movimiento se puede evaluar psíquicamente a una persona, conocer rasgos de su personalidad ó carácter, y también como pueden ser modificados si se corrigen estas pautas. Con este sistema se consigue un análisis del movimiento del sujeto sin la necesidad de un especialista, lo que supone un gran avance. Un sistema cuya información de entrada es el modelo cinemático y cuya salida corresponde a los esfuerzos lo que permite identificar indirectamente al menos las ocho acciones básicas del esfuerzo.

Capítulo 3

Reconocimiento del gesto

3.1 Reconocimiento de patrones temporales. Redes neuronales.

Desde la primera mitad del siglo XX se ha intentado emular el comportamiento del cerebro humano mediante modelos computacionales. Uno de estos modelos son las redes neuronales artificiales (Gestal).

Según (Haykin, 1999):

"*Una red neuronal es un procesador distribuido y con una estructura paralela que tiene una tendencia natural a almacenar conocimiento experimental, haciéndolo apto para su uso. Se parece al cerebro en dos cosas:*

- *El conocimiento es adquirido por la red a través de un proceso de aprendizaje*
- *Ese conocimiento se almacena en los pesos sinápticos o conexiones entre neuronas*"

Las redes neuronales se pueden englobar en un campo de la computación que emula sistemas biológicos destinado a resolver problemas que resultarían complicados de analizar con un algoritmo tradicional, tales como extracción de patrones, detección de tramas, toma de decisiones, etc., gracias a esto y su facilidad de uso se han hecho muy populares. Su aprendizaje adaptativo, capacidad de auto-organización, tolerancia a fallos, operación en tiempo real y fácil inserción dentro de la tecnología existente, han hecho que su utilización se haya extendido en áreas como la biología, financiera, salud, industrial, medioambiental, etc.

Las redes neuronales también tienen inconvenientes, por ejemplo, debido al complejo procesamiento que realizan no es posible seguir *paso a paso* el razonamiento que les ha llevado a extraer sus conclusiones.

En general una red se suele caracterizar por tres partes fundamentales: la *topología de la red*, las *reglas de aprendizaje* y el *tipo de entrenamiento*. En este proyecto se opta por una red neuronal para procesar los datos por su capacidad de reconocimiento de patrones temporales. Los datos de entrada a la red varían con el tiempo y se necesita un sistema robusto capaz de reconocerlos y clasificarlos.

3.1.1 Un poco de historia

Las redes neuronales tal y como se conocen hoy día datan de 1943, año en el que McCulloch y Pitts, pioneros en este campo, publicaron un trabajo que describía el cálculo

lógico de una red neuronal, en la cual las neuronas que la componían seguían una regla de "todo o nada". Con este estudio sentaban las bases de la Inteligencia Artificial.

En 1958 Rosenblatt hizo un acercamiento al problema de reconocimiento de patrones con el *Perceptrón*, un intento de entender la memoria humana, el aprendizaje y los procesos cognitivos. Mark I, como fue denominado el perceptrón, estaba compuesto por un número determinado de entradas unidas por unas conexiones ponderadas a una única neurona de salida, para que esta neurona cambiase de estado la suma de las entradas ponderadas debía de superar un umbral.

En 1960 Widrow y Hoff introdujeron el algoritmo del Error Cuadrático Medio Mínimo (LMS, *Least Mean Squared Error*) ó Regla Delta con el cual formularon el *Adaline* (ADAptiveLINearElement). La principal diferencia entre el perceptrón y el Adaline reside en el proceso de entrenamiento. También fueron ellos quienes desarrollaron *Madaline* (Multiple-Adaline), que como su propio nombre indica, estaba compuesta por varias estructuras *Adaline* en paralelo con lo que conseguían múltiples entradas y una salida. Hasta ahora con estos sistemas sólo se podían resolver problemas linealmente separables, pero con un conjunto de Madaline en paralelo se resolvía este inconveniente. Las redes neuronales tienen aplicación en el campo del procesamiento de señales. Se utilizan para la implementación de filtros que eliminan el ruido de señales portadoras de información, la cancelación del ruido materno en grabaciones ECG del latido del feto humano ó para construir filtros de ecualización adaptativos en módems de alta velocidad.

En los sucesivos años que siguieron a esta innovación, los investigadores lograron múltiples avances dando lugar a estructuras utilizadas hoy en día: Red de Hopfield (1982), Red de Kohonen (1982) y la Máquina de Boltzmann (1985). Actualmente su uso está muy extendido. Existen aplicaciones capaces de realizar reconocimiento facial, compresión de imágenes, traducción en tiempo real, análisis de células cancerígenas, diseño de prótesis, predicción de índices bursátiles, eliminación de ruido, etc.

3.1.2 Modelo biológico y artificial

En nuestro sistema nervioso existen células llamadas neuronas que son verdaderas unidades de procesamiento. Hay cerca de 10 billones de neuronas en el córtex y 60 trillones de sinapsis. Dando como resultado una estructura muy eficiente. Las sinapsis son elementos funcionales y estructurales que median en la interacción entre neuronas e imponen un estado de excitación o inhibición, nunca ambos a la vez, en la neurona receptora. Gracias a esta fuerza de interconexión entre las neuronas la red puede ser capaz de cómputos y detección de patrones complejos. La más común es la sinapsis química. El estímulo eléctrico recibido al pasar de un cierto umbral causa que la neurona, a su vez, envíe una señal eléctrica a través de su axón a otras sucesivas neuronas. Los axones son líneas de transmisión y las dendritas zonas receptoras.

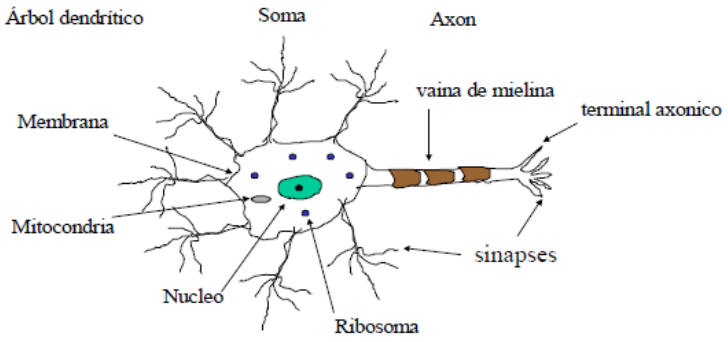


Figura 5: Representación de una neurona biológica.[6]

La neurona artificial al igual que la biológica es un elemento sencillo de procesado, las sinapsis (uniones entre neuronas) se caracterizan por la ponderación de sus pesos sinápticos, todas las entradas que llegan a dicha neurona son ponderadas por estos pesos y sumadas. Esta suma alimenta la función de activación cuya misión es acotar la salida a un cierto rango de valores. Es muy común el uso de una entrada con un valor fijo, *bias*, la cual es ponderada y sumada al resto de los pesos incrementando o disminuyendo el valor de la suma que deberá llegar a la función de activación

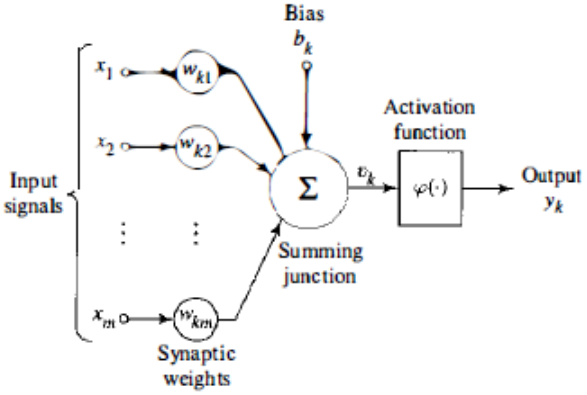


Figura 7: Procesado de datos en una neurona artificial.

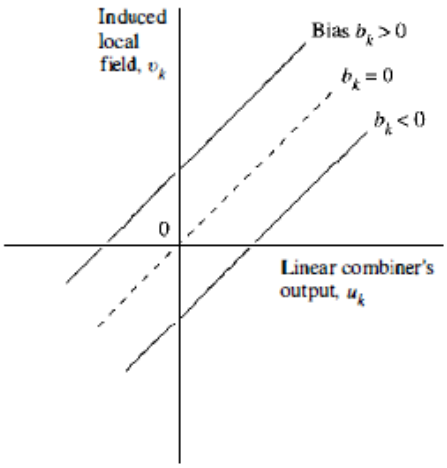


Figura 6: Efecto del uso de *bias* en la salida.

$$v_k = \sum_{j=1}^{m} w_{kj} * x_j + b_k; \quad (1)$$

$$\boldsymbol{y_k} = \varphi(v_k); \quad (2)$$

[6] Introducción a las redes neuronales. Tomás Arredondo Vidal. Depto. Electrónica UTFSM.

3.1.3 Problemas adecuados para resolución con redes

Las redes neuronales no son adecuadas para resolver todo tipo de problemas (Heaton J. , 2005). Un buen ejemplo de un programa que no es adecuado para la resolución mediante una red neuronal es aquel que se puede escribir fácilmente como diagrama de flujo, es decir, que conste de pasos a seguir bien definidos.

Otro criterio a tomar en consideración es la probabilidad de que la lógica del programa cambie. La habilidad de aprender es una de las principales características de una red neuronal por lo que si el algoritmo utilizado para resolver un problema es una regla fija podría ser perjudicial, ya que la red tenderá a encontrar una solución adecuada y comenzará a divergir de la solución esperada.

Por último las redes no son adecuadas para problemas en los cuales se ha de saber exactamente como se ha llegado a la solución. Una red puede ser una experta resolviendo el problema para el cual se le ha entrenado pero no puede explicar los pasos que ha seguido hasta llegar a esa respuesta.

3.1.4 Estructuras

Se puede hacer una clasificación en tres grandes tipos de estructuras:

3.1.4.1 Redes de una capa

Este tipo de redes están formadas por una capa de entrada y una de salida. La señal siempre viaja en un sólo sentido, desde la capa de entrada a la de salida (*feedforward propagation*). Se dice que están formadas por una sola capa porque a la hora de realizar los cálculos sólo entra en juego la capa de salida, la capa de entrada se encarga de presentarles los patrones. Véase Figura 12. A continuación se hace una demostración del proceso de cálculo que sigue una red neuronal para una mejor comprensión del lector.

En una red de una sola capa formada por 4 neuronas de entrada y 4 de salida, y con 3 conjuntos m de entradas-salidas, el desarrollo es el siguiente:

- Cada conjunto m está formado por una entrada x_j y su correspondiente salida esperada y_j, que se utiliza en el cálculo del error que se explicará más adelante, para este caso concreto son:
  $m_1 = (x_{11}x_{12}x_{13}x_{14}, y_{11}y_{12}y_{13}y_{14})$;
  $m_2 = (x_{21}x_{22}x_{23}x_{24}, y_{21}y_{22}y_{23}y_{24})$;
  $m_3 = (x_{31}x_{32}x_{33}x_{34}, y_{31}y_{32}y_{33}y_{34})$.

- Cada neurona de entrada tiene una conexión con una cada una de las neuronas de salida, por lo que el número de conexiones entre neuronas es de 16, y todas ellas ponderadas por un peso sináptico w_{kj} .

- Con los datos anteriores y sabiendo que cada neurona de salida tiene una función de activación la salida queda representada por:

$$y_{11} = \varphi(x_{11} * w_{11} + x_{12} * w_{12} + x_{13} * w_{13} + x_{14} * w_{14} + bias * w_{15});$$

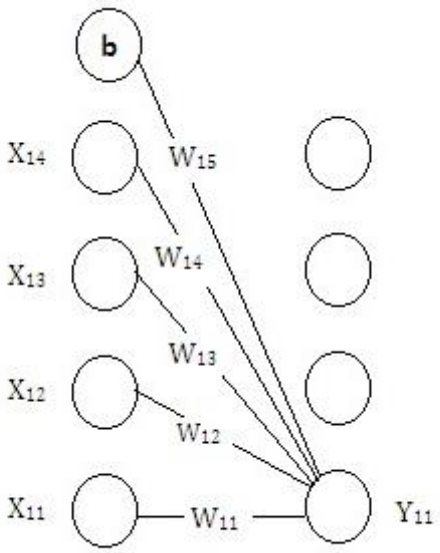


Figura 8: Proceso de cálculo de red neuronal de una capa I.

$$y_{12} = \varphi(x_{11} * w_{21} + x_{12} * w_{22} + x_{13} * w_{23} + x_{14} * w_{24} + bias * w_{25});$$

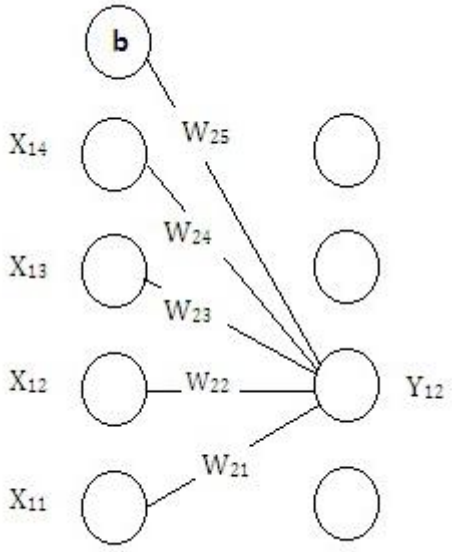


Figura 9: Proceso de cálculo de red neuronal de una capa II.

$$y_{13} = \varphi(x_{11} * w_{31} + x_{12} * w_{32} + x_{13} * w_{33} + x_{14} * w_{34} + bias * w_{35});$$

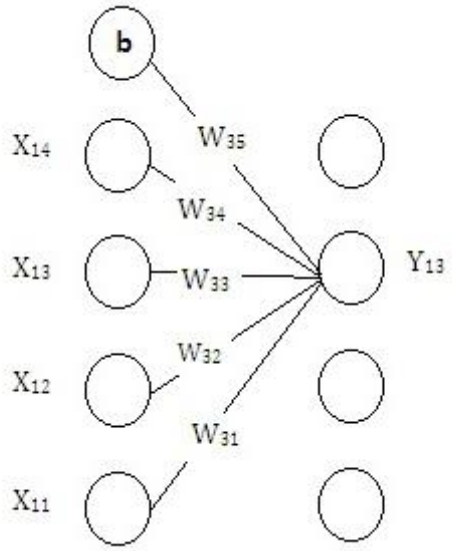


Figura 10: Proceso de cálculo de red neuronal de una capa III.

$$y_{14} = \varphi(x_{11} * w_{41} + x_{12} * w_{42} + x_{13} * w_{43} + x_{14} * w_{44} + bias * w_{45});$$

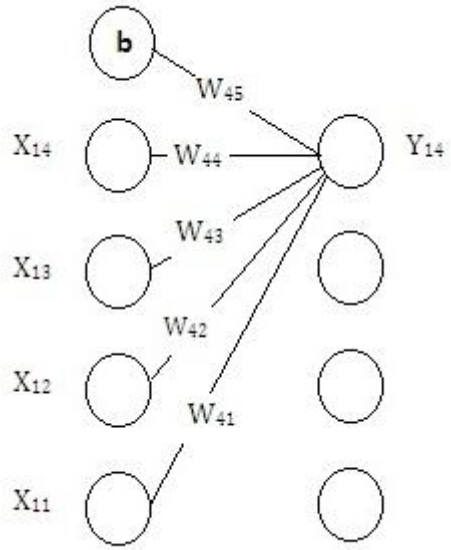


Figura 11: Proceso de cálculo de red neuronal de una capa IV.

- El proceso para los conjuntos m_2 y m_3 es el mismo.

3.1.4.2 Redes multicapa

Estas redes se caracterizan por tener capas ocultas entre las de entrada y salida que actúan de enlace. La señal se propaga en una sola dirección (*feedforward*). En dependencia del número de capas ocultas podemos obtener mayor o menor exactitud en los resultados, que también mejorarán cuanto mayor sea el tamaño de esta capa. El modo de funcionamiento es el siguiente: se presentan los patrones a la capa de entrada que a su vez entrega la señal a la capa oculta, estas neuronas computan la señal y entregan su salida a otra capa oculta o a la capa de salida que volverán a computar la señal y producirán otra salida. La salida de la última capa es la respuesta de la red al patrón que hayamos presentado a su entrada. Véase Figura 13.

El cálculo en este tipo de redes es igual que en las de una sola capa, con la única diferencia de que una vez se tienen los cálculos de la primera capa esas salidas intermedias son las entradas de la siguiente capa, así hasta obtener la salida total de la red.

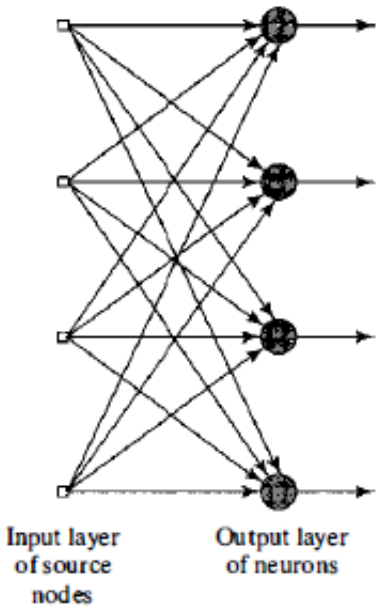


Figura 12: Red neuronal de una capa.

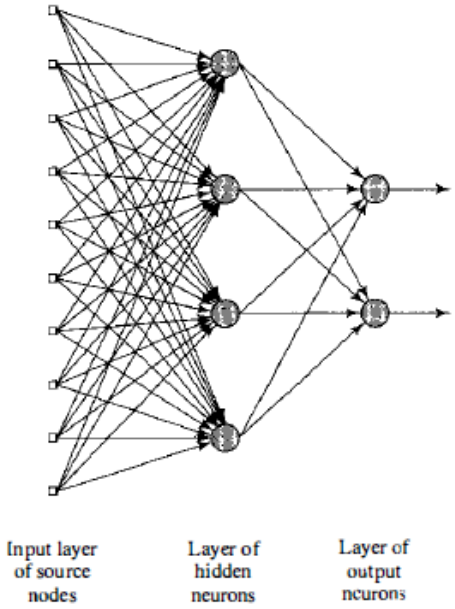


Figura 13: Red neurona multicapa.

3.1.4.3 Redes recurrentes

Estas redes se caracterizan por tener al menos un bucle de retroalimentación, es decir, una conexión en la salida de una de las capas y realimentar su entrada. Las neuronas pueden ser auto-alimentadas con su propia señal de salida o con las de otras.

El uso de estos bucles tiene un gran impacto sobre la capacidad de aprendizaje y el funcionamiento de la red, este uso implica la utilización de ramas con elementos de retardo (z^{-1}), que resultan en un comportamiento dinámico no-lineal. Véase Figura 14.

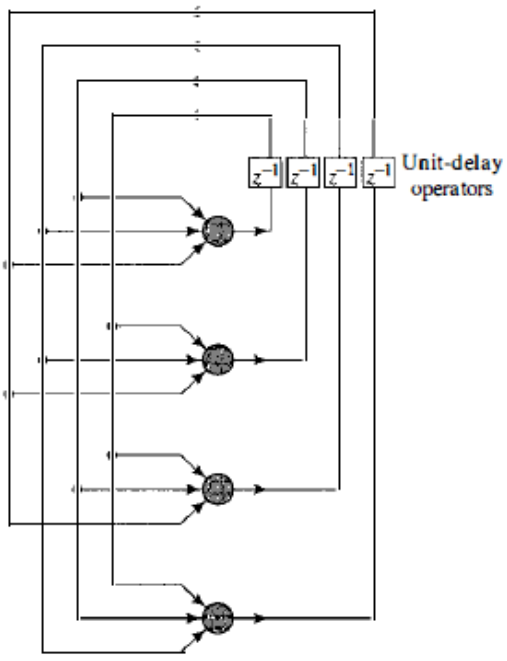


Figura 14: Red neuronal recurrente.

3.1.5 Funciones de activación

La función de activación limita la amplitud de la salida de una neurona a un intervalo de [0,1] ó [-1,1]. Estas nos permitan dar más flexibilidad al sistema y dotarlo de más capacidad.

Dependiendo de las salidas que queramos obtener deberemos usar unas u otras, para rangos de salida positivos usaremos la *función escalón*, la *función lineal* y la *función sigmoide*, para salidas positivas y negativas usaremos la *función de tangente hiperbólica*.

Cabe destacar que la característica más importante de la función de activación es su no linealidad, permitiendo así tratar problemas en los que la relación entre las entradas y las salidas de un sistema no tienen relación, diferenciándolas de otros medios de procesado de información.

3.1.5.1 Función escalón

En este modelo la salida de una neurona toma el valor de 1 si el campo local inducido es mayor que 0 y 0 si toma valores negativos.

$$y_k = \begin{cases} 1 \; si v_k \geq 0 \\ 0 \; si v_k \leq 1 \end{cases} \tag{3}$$

$$v_k = \sum_{j=1}^{m} w_{kj} * x_j + b_k \text{x} \tag{4}$$

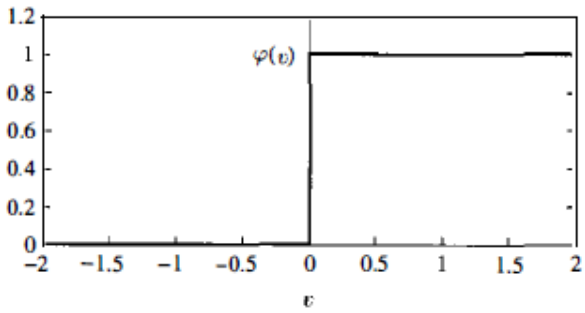


Figura 15: Función escalón

3.1.5.2 Función lineal

La salida de esta función de activación se aproxima al funcionamiento de un amplificador no lineal. El factor de amplificación en la zona lineal se considera 1.

$$\varphi(v) = \begin{cases} 1 \;\; \boldsymbol{si} \; v \geq \frac{1}{2} \\ v \;\; \boldsymbol{si} \;\; -\frac{1}{2} \geq v \geq \frac{1}{2} \\ 0 \;\; \boldsymbol{si} \; v \leq \frac{1}{2} \end{cases} \tag{5}$$

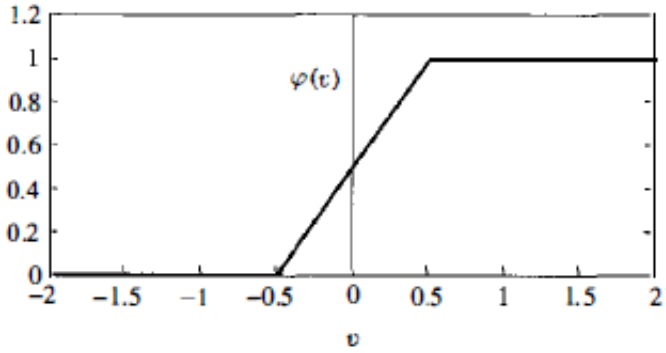


Figura 16: Función lineal

Tiene dos modos de actuación dependiendo del factor de amplificación, si este se hace infinitamente grande actúa como la función escalón.

3.1.5.3 Función sigmoide

Es la más utilizada. Se define como una función estrictamente creciente que tiene un comportamiento entre lo lineal y lo no lineal.

$$\varphi(v) = \frac{1}{1+e^{-av}} \tag{6}$$

a: Determina la pendiente.

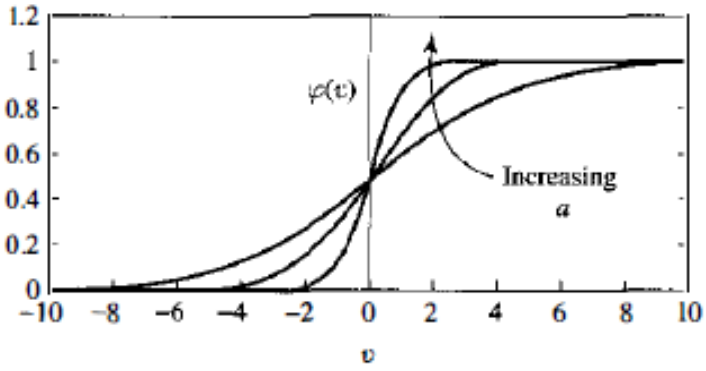


Figura 17: Función sigmoide.

La pendiente en el origen es $a = \frac{3}{4}$, en límite se aproxima a 1 y la sigmoide se convierte en función escalón, con la diferencia de que la sigmoide toma valores continuos entre 0 y 1.

3.1.5.4 Función tangente hiperbólica

Esta función nos proporciona salidas negativas cosa que las anteriores no hacían.

$$\varphi(v) = \begin{cases} 1 \; \boldsymbol{si} \; v > 0 \\ 0 \; \boldsymbol{si} \; v = 0 \\ -1 \; \boldsymbol{si} \; v < 0 \end{cases} \tag{7}$$

$$\tan h\,(v) = \frac{e^{v} - e^{-v}}{e^{v} + e^{-v}} \tag{8}$$

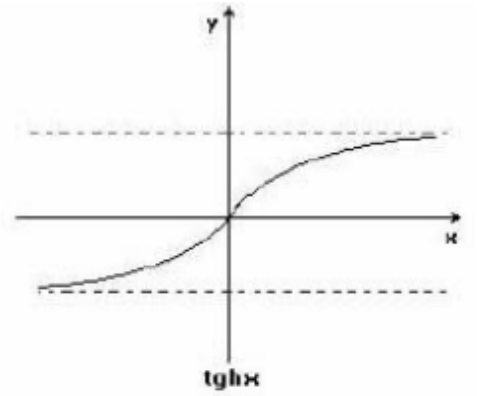


Figura 18: Función hiperbólica.

3.1.6 Tipos de aprendizaje

No hay un único tipo de aprendizaje, existe una gran variedad de algoritmos de aprendizaje, cada uno con sus ventajas y desventajas.

La principal diferencia entre ellos es la manera en la que cada uno ajusta los pesos sinápticos. A continuación se describen algunos de los más utilizados: *corrección de error*, *aprendizaje basado en instancias* y *competitivo*.

3.1.6.1 Aprendizaje por corrección de error

Teniendo:

$$e_j(n) = d_j\,(n) - y_j(n); \tag{9}$$

Donde:

$e_j(n)$: Señal de error en neurona de salida j para iteración n.
$d_j(n)$: Respuesta esperada para la entrada presentada.
$y_j(n)$: Señal actual obtenida a la salida.

La señal de error e_j actúa como un mecanismo de control para el ajuste de los pesos sinápticos. Los ajustes, a través de la minimización de la función de coste, hacen que la señal y_j se vaya acercando a y_j. Este proceso continúa hasta que los pesos alcanzan un valor estable.

3.1.6.2 Aprendizaje basado en instancias

Este algoritmo en lugar de hacer generalizaciones explícitas, compara el problema que se le presenta, x_{test}, con ejemplos ya vistos durante el entrenamiento. Recupera y analiza información del entrenamiento en un "vecindario local" de x_{test}, que ha sido almacenado en la memoria y lo clasifica comparándolo con este.

Los datos se almacenan en un mismo vector:

$\{(x_i, d_i)\}_{i=1}^{N}$

x_i:Vector de entrada.
d_i:Respuesta deseada.

3.1.6.3 Aprendizaje competitivo

En este caso las neuronas de salida compiten entre ellas para ser activadas, sólo puede ser activada una neurona a la vez. Para que una neurona sea la ganadora, la suma de todos los inputs que llegan a esta por los pesos sinápticos correspondientes, debe ser la mayor de todas las neuronas que compiten. En ese caso la salida de la neurona ganadora se iguala a 1 y las de las demás a 0. Los pesos sinápticos sólo se ajustan para la neurona ganadora:

$$\Delta_{w_{kj}} = \begin{cases} (x_j - w_{kj}) & Neurona\ \ ganadora \\ 0 & Resto \end{cases} \tag{10}$$

3.1.6.4 Aprendizaje por refuerzo

Es un aprendizaje supervisado más lento que el anterior. En este caso lo único que se proporciona a la red es un refuerzo para indicarle si se ha ajustado a la salida deseada o no y en función de esto se ajustan pesos y *bias*.

3.1.7 Tipos de entrenamiento

3.1.7.1 Entrenamiento supervisado

Para este tipo de entrenamiento se han de proveer a la red tanto las entradas como las salidas correctas para esas entradas. Los parámetros de la red son ajustados en función del error obtenido a la salida. En cada iteración el error debe hacerse más pequeño aproximándose el valor de la salida a la salida correcta.

Cuando en una red se tienen varias salidas se deberán presentar los juegos de ensayo uno por uno e ir ajustando los parámetros en función de la entrada en ese momento hasta completar todos los que queramos que aprenda y volver a empezar hasta que la red sea capaz de reconocer todas las entradas. Es el tipo de entrenamiento más común.

Unos ejemplos de este tipo de redes son: Perceptrón simple y Adaline.

Algoritmos de entrenamiento

Existen múltiples algoritmos de entrenamiento, en este proyecto vamos a centrarnos en un algoritmo para entrenamientos supervisados: backpropagation o retropropagación.

Backpropagation

Este método de entrenamiento es sencillo y uno de los más utilizados, se van presentando patrones de uno en uno a la red, tanto entradas como salidas, hasta completar el juego entero de entrenamiento. El ajuste de los pesos se hace de acuerdo a los errores presentados en cada patrón (entrenamiento supervisado).

Se procede de la siguiente manera, se presentan los datos de entrada a las neuronas de la primera capa y se hace un pase hacia delante (*feed-forward*), se calculan las salidas de las neuronas según se va pasando de una capa a la siguiente. En la última capa, se compara la salida que se obtiene en cada neurona con la salida que se esperaba obtener y en función de ese error se ajustan los pesos y el *bias* de la red.

La forma de calcular el error se seleccionará en dependencia del problema que se quiera resolver, siendo los más comunes el *Error Cuadrático Medio Mínimo* (*LMS*) para problemas de regresión lineal y la *Función de Error de Entropía Cruzada*, regresión logística, para problemas de clasificación.

A continuación se detalla cómo actúa el algoritmo *Backpropagation* frente a problemas de clasificación. Primero aclarar que el análisis que se realiza con la regresión logística, se utiliza para predecir el resultado de una variable categórica en función de las variables independientes (Hosmer & Lemeshow, 2000) .Correlaciona la probabilidad de una variable binaria, 0 ó 1, con una variable escalar x. En este caso la probabilidad aproximada del suceso se aproxima mediante una función logística:

$$h_\Theta = \frac{1}{1+e^{-\theta x}} \; ; \qquad (11)$$

El objetivo de este algoritmo es minimizar la función de coste J(Θ) que depende de la hipótesis $h_\Theta(x)$.Cuando presentamos una entrada a la red obtenemos una salida $h_\Theta(x)$ que depende de la entrada x. Mediante la modificación de los pesos sinápticos Θ en cada iteración se consigue que $h_\Theta(x)$ sea lo más cercana posible a $y(x)$ y así minimizar J(Θ) .

$$\mathrm{J}(\Theta) = -\frac{1}{m}\sum_{j=1}^{m}[y_i * \log h_\Theta(x_i) + (1-y_i) * \log(1-h_\Theta(x_i))] + \frac{\lambda}{2m}\sum_{j=1}^{n}\Theta_j^2; \quad (12)$$

m: Número de conjuntos entradas-salidas que se presentan a la red.
x_i: Entrada i del conjunto m.
y_i: Salida i del conjunto m.
$h_\Theta(x_i)$: Hipótesis obtenida en función de x_i.
λ: Parámetro de regularización.

Θ_j : Pesos sinápticos.

Con el término de regularización: $\frac{\lambda}{2m}\sum_{j=1}^{n}\Theta_j^2$, se controla el *overfitting*[7].

Se elige esta función de coste porque puede derivar de la estadística usando el principio de Estimación de Máxima Verosimilitud (*Maximum Likelihood Estimation*), lo que significa que hay una distribución Gaussiana en la relación de sus parámetros, además de ser una función convexa (Ng, 2011).

Una vez se conoce el error se ajustan los pesos y el *bias* en la función de éste. Primero se hace un pase hacia atrás (*backwards*) y se calcula el error que aporta cada neurona a la salida final mediante las deltas, a continuación se regularizan y mediante el algoritmo de gradiente descendente se actualizan los pesos de la red.

Proceso de ajuste de los pesos sinápticos

El valor del error δ_j en la capa de salida es:

$$\delta_j = d_j - y_j; \quad (13)$$

d_j : Salida deseada.
y_j : Salida obtenida.

Si la neurona pertenece a la capa oculta al no tener una salida esperada su error se calcula en función de los errores de las neuronas a las que están conectadas.

El valor del error δ_j definido como:

$$\delta_j = \varphi'_j(v_j) * \sum_k \delta_k * \Theta_{kj}; \quad (14)$$

δ_k : Valor del error de las neuronas de la capa anterior k que conectan con la neurona j.
$\Theta_{kj}(n)$: Pesos sinápticos que unen a las neuronas de la capa j y k.

El siguiente paso es calcular el gradiente, la derivada parcial de la función de coste respecto de los pesos sinápticos $\frac{\partial J(\Theta)}{\partial \Theta_j}$.Hay que distinguir dos casos, cuando el cálculo pertenece a un *bias* o no:

$\frac{\partial J(\Theta)}{\partial \Theta_j} = \frac{1}{m}\Delta_{ij} + \lambda\Theta_{ij}$. Si se calcula para una neurona cualquiera. (15)

$\frac{\partial J(\Theta)}{\partial \Theta_j} = \frac{1}{m}\Delta_{ij}$. Si se calcula para *bias*. (16)

$\Delta_{ij} = \Delta_{ij} + v_j * \delta_j;$

[7] El *Overfitting* o sobreentrenamiento se produce cuando la red no es capaz de extrapolar su conocimiento ante nuevas entradas porque ha "aprendido" demasiado bien las muestras con las que ha sido entrenadas.

Por último la corrección que se aplica al peso sinápticoΘ_j es proporcional a la derivada parcial de la función de coste respecto de los pesos sinápticos $\frac{\partial J(\Theta)}{\partial \Theta_j}$.

$$\Theta_j = \Theta_j - \alpha \frac{\partial J(\Theta)}{\partial \Theta_j}. \quad (17)$$

El signo menos indica la dirección en que la pendiente decrece, según se vaya aproximando a un mínimo, el algoritmo irá tomando pasos más pequeños. Si $J(\Theta)$ decrece menos de 10^{-3} en una iteración podemos declarar su convergencia.

A: es la tasa de aprendizaje, *learning rate*. Controla la rapidez con la que converge el algoritmo. Si α es muy pequeño la convergencia puede ser muy lenta, por otro lado, si α es demasiado grande puede no llegar a converger ó incluso divergir.

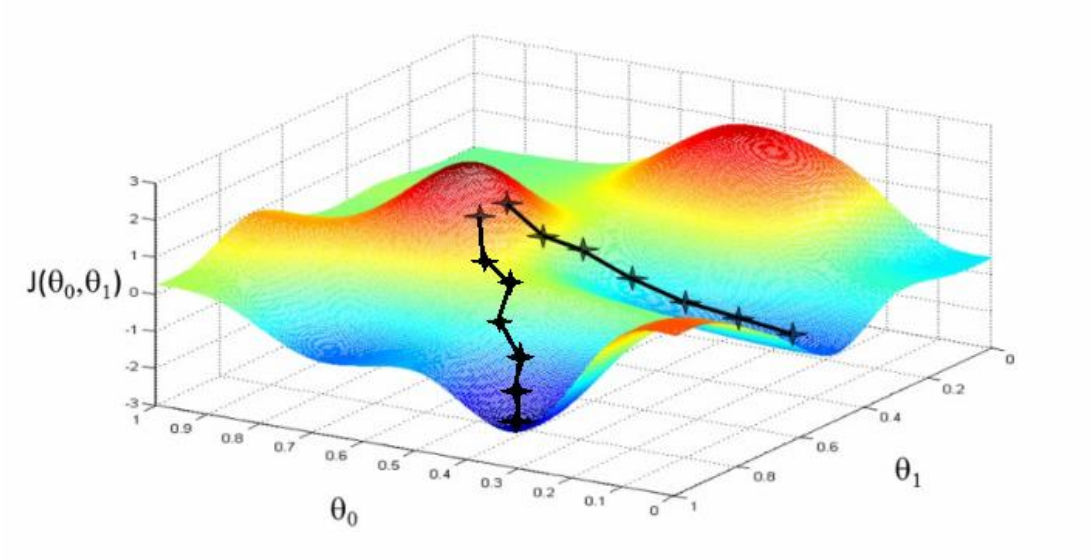


Figura 19: Gráfica de mínimos local y global. Proceso de descenso de gradiente.[8]

Un inconveniente del gradiente descendente que hay que tener en cuenta es la posibilidad de que el algoritmo caiga en un mínimo local. En dependencia de la inicialización de los pesos se inicia en un punto u otro de la función de coste, cuando se cae en un mínimo la pendiente es igual a 0, $\frac{\partial J(\Theta)}{\partial \Theta_j} = 0$, por lo que Θ_j no se actualiza.

3.1.7.2 Entrenamiento no supervisado

Es parecido al entrenamiento supervisado pero sin proporcionar las salidas. Se utiliza cuando el cometido de la red es clasificar las entradas en varios grupos. Con el progreso del entrenamiento la red va descubriendo los grupos en los que ha de clasificar los resultados. Unos ejemplos de este tipo de redes son: Las redes de aprendizaje competitivo, las redes de Hopfield ó la máquina de Boltzmann (Heaton J. , 2005).

[8]http://blog.datumbox.com/tuning-the-learning-rate-in-gradient-descent/

3.1.8 Pautas para el diseño de una red

3.1.8.1 Elección del número de capas y neuronas

El número de neuronas en la capa de entrada está determinado por el número de variables de entrada, al igual que el número de neuronas en la capa de salida que es igual al número de variables que se quieran obtener.

La elección del número de capas intermedias depende de la complejidad del problema, siendo suficiente en la mayoría de ellos una sola capa, con dos capas se obtienen mejores resultados aunque el coste computacional es mayor, no se suelen utilizar redes de más de capas ocultas.

El número de neuronas en las capas ocultas se elige de forma tal que no sean ni menor que el número de neuronas de la capa de salida ni mayor que el número de neuronas de la capa de entrada. Generalmente siguiendo un orden piramidal descendente (Heaton J. , 2005).

3.1.8.2 Inicialización de los pesos

Los pesos controlan el comportamiento dinámico de la red, se eligen entre [-1,1], es decir $|w|<1$, para que el sistema sea estable, esto corresponde a un sistema con memoria infinita, en el sentido de que la salida depende de las muestras de entrada, que se extienden hasta el infinito. Para estos pesos la señal de salida es exponencialmente convergente lo que quiere decir que el sistema es estable. La red debe ejecutarse varias veces con diferentes inicializaciones de los pesos para poder comparar si alguno de los resultados obtenidos era un mínimo local y no el global.

3.1.9 Normalización

Para una mayor eficiencia en los resultados se deben normalizar las entradas antes de alimentar la red. El rango en el que se encuentren dependerá de la función de activación que se vaya a utilizar.

3.1.10 Entrenamiento de la red y error

Para comenzar a probarla red se debe tener un juego de ensayos. En la fase de entrenamiento se deben presentar a la red patrones de estímulos de entrada del juego de ensayos. El juego de ensayo se debe elegir cuidadosamente para que quede representada toda la información que la red necesita aprender. Normalmente consta del 60% de las muestras totales. Una vez ajustados los pesos y el *bias* en función de estas entradas, se presenta a la red el siguiente grupo que consta del 20% de las muestras, este grupo se denomina de *validación cruzada*. Comparando la función de coste obtenida en este grupo con la obtenida en el grupo de entrenamiento podemos determinar si la

red está *sobreentrenada* (*high variance*) ó por el contrario el entrenamiento ha sido pobre (*high bias*).

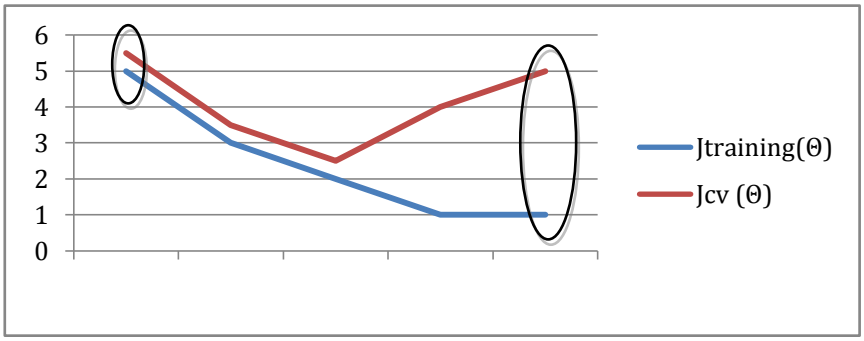


Figura 20: Gráfica *High variance* vs. *High bias*

Como se puede apreciar en el gráfico cuando los dos errores son altos se tiene un problema de *high bias*, es decir, el entrenamiento ha sido pobre; la red no es capaz de identificar los patrones que se le presentan. En el lado izquierdo, sin embargo, el error correspondiente al conjunto de entrenamiento es bajo y para el conjunto de validación aumenta, en este caso la red está sobreentrenada y no es capaz de extrapolar características de las nuevas entradas que se le presentan. Lo ideal es encontrar el punto en el que los dos errores son semejantes y aceptables, esto se consigue mediante la técnica de *early stopping* en la cual se detiene el proceso del conjunto de validación en cuanto el error comienza a crecer; sirve para evitar el sobreentrenamiento (*overfitting*).

Para controlar este inconveniente también se dibujan las curvas de aprendizaje. Las curvas de aprendizaje muestran como la red mejora según aumenta el número de iteraciones. De nuevo se comparan las funciones de coste obtenidas durante el entrenamiento y con el conjunto de validación pero esta vez frente al número de iteraciones.

Por último se presenta a la red el grupo denominado de prueba, este grupo contiene el 20% de las muestras restantes. Con este grupo se evalúa el rendimiento final de la red.

3.1.11 Criterios de finalización del entrenamiento

Debido a que no siempre se alcanza el mínimo deseado de la función de error es necesario imponer otras pautas para que la red cese en su entrenamiento:

- Número máximo de iteraciones.
- Umbral mínimo de error.
- Número máximo de iteraciones sin que el error disminuya.

3.2. Captura del gesto

Se han seleccionado 8 movimientos, cada unos de los cuales caracteriza una polaridad de subcategoría del *esfuerzo*.

Subcategoría del esfuerzo	**Polaridad**	**Tipo de movimiento**
Espacio	Directo	Enhebrar una aguja
	Indirecto	Apartar bichos
Peso	Pesado	Empujar un objeto pesado
	Liviano	Pintar un lienzo
Tiempo	Sostenido	Bostezar
	Súbito	Saltar para coger una pelota
Flujo	Guiado	Llevar una taza de té caliente
	Libre	Tirar una piedra a un estanque

Tabla 2: Movimientos asociados a las subcategorías del *esfuerzo*.

También se prueba la red para movimientos complejos. Estos movimientos son la combinación de dos polaridades de subcategoría de esfuerzo diferentes ya que no existen movimientos que puedan clasificarse en dos polaridades opuestas al mismo tiempo.

Nº de combinación	**Subcategoría del esfuerzo**	**Polaridad**	**Tipo de movimiento**
1	Espacio	Indirecto	Apartar bichos
	Peso	Pesado	Empujar un objeto pesado
2	Espacio	Directo	Enhebrar una aguja
	Peso	Liviano	Pintar un lienzo
3	Tiempo	Sostenido	Bostezar
	Flujo	Guiado	Llevar una taza de té caliente
4	Tiempo	Súbito	Saltar para coger una pelota
	Flujo	Libre	Tirar una piedra a un estanque

Tabla 3: Combinación de movimientos

Para poder configurar los diferentes conjuntos de prueba, entrenamiento, validación cruzada y prueba, se realiza cada uno de los movimientos un número determinado de veces delante de la cámara kinect. Esto da como resultado cada uno de ellos un base de datos (archivo de formato *csv*[9]) con sus datos de velocidad y aceleración de cada articulación.

[9]Comma Separated Value

3.3. Procesamiento del gesto

La red es estática, no tiene realimentación, por lo que se le pasan valores fijos que no varían con el tiempo en el momento del procesamiento. Para la obtención de estos valores se hace una media de cada archivo *csv* con el programa "*media.pde*", para cada archivo quedan 30 valores: 15 de velocidad y 15 de aceleración.

Con las pautas indicadas anteriormente se configuran los conjuntos de entrenamiento, validación cruzada y prueba. Con estos juegos es posible iniciar el entrenamiento de la red. La capa de entrada consta de 30 neuronas, que son los datos en la entrada y la capa de salida de 8 neuronas, lo que coincide con cada polaridad. La elección del número de neuronas de las capas ocultas dependerá de los resultados que se obtengan y variarán en función de estos. Otros parámetros a tener en cuenta son α, λ, $J(\Theta)$ y el número de iteraciones, que también cambiarán sus valores según el rendimiento de la red.

3.4. Clasificación del gesto

Para considerar que la red funciona, deben activarse la(s) neurona(s) que correspondan al movimiento que se le está mostrando. Una neurona se considera activa cuando su salida es la más cercana a 1.

Subcategoría del esfuerzo	**Polaridad**	**Tipo de movimiento**	**Resultado de salida de la red**
Espacio	Directo	Enhebrar una aguja	(1,0,0,0,0,0,0,0)
	Indirecto	Apartar bichos	(0,1,0,0,0,0,0,0)
Peso	Pesado	Empujar un objeto pesado	(0,0,1,0,0,0,0,0)
	Liviano	Pintar un lienzo	(0,0,0,1,0,0,0,0)
Tiempo	Sostenido	Bostezar	(0,0,0,0,1,0,0,0)
	Súbito	Saltar para coger una pelota	(0,0,0,0,0,1,0,0)
Flujo	Guiado	Llevar una taza de té caliente	(0,0,0,0,0,0,1,0)
	Libre	Tirar una piedra a un estanque	(0,0,0,0,0,0,0,1)

Tabla 4: Resultado a la salida de la red para movimientos simples.

Nº de combinación	**Subcategoría del esfuerzo**	**Polaridad**	**Tipo de movimiento**	**Resultado de salida de la red**
1	Espacio	Indirecto	Apartar bichos	(1,0,0,1,0,0,0,0)
	Peso	Pesado	Empujar un objeto pesado	
2	Espacio	Directo	Enhebrar una aguja	(0,1,1,0,0,0,0,0)
	Peso	Liviano	Pintar un lienzo	
3	Tiempo	Sostenido	Bostezar	(0,0,0,0,1,0,0,1)
	Flujo	Guiado	Llevar una taza de té caliente	
4	Tiempo	Súbito	Saltar para coger una pelota	(0,0,0,0,0,1,1,0)
	Flujo	Libre	Tirar una piedra a un estanque	

Tabla 5: Resultado a la salida de la red para movimientos complejo.

Capítulo 4

Experimentación

En este capítulo se trata la programación de la red y sus diferentes funciones.

4.1 Redes

La red ha sido creada con la librería **Encog 3.1.0** a excepción de algunas funciones de las que carecía; que se han programado concretamente para este proyecto pero antes de describir los bloques principales de la red se va a comentar el uso de programas auxiliares que se han utilizado en la consecución de los objetivos.

Sobre el programa principal del que se obtienen las variables cinemáticas se han incorporado dos modificaciones, la primera es la grabación en vídeo del movimiento en cuestión y la segunda la exportación a un fichero *csv* de los valores grabados en el vídeo. El código de grabación se encuentra en el archivo "*Recorderplay.pde*"[10] y se adapta para su uso en este programa, mientras que el archivo *csv* proviene de un *JSON*[11], que es el formato en el que el programa original almacena los datos. Sólo se incluyen los cambios realizados por la autora de este proyecto sobre el código original. Véase Código 1.

Figura 21: Fotograma de una captura con kinect.

[10] https://code.google.com/p/simple-openni/source/browse/trunk/SimpleOpenNI/dist/all/SimpleOpenNI/examples/OpenNI/RecorderPlay/RecorderPlay.pde?r=382
[11] JavaScript Object Notation

```
String [] jointArray= new String [15];
float [] aceleracionArray = new float [15];
float [] velocidadArray = new float [15];

float [] acele= new float [15];
float [] veloci= new float [15];

Table table;


  public void run() {
    JSONObject datoSalida = new JSONObject();

      table = new Table();

      table.addColumn("JOINT");
      table.addColumn("VELOCIDAD", Table.FLOAT);
      table.addColumn("ACELERACIÓN", Table.FLOAT);
      table.addColumn("SEGUNDOS", Table.FLOAT);


      datoSalida = empaquetarDatos();
      enviarDatos(datoSalida);

      saveJSONObject(datoSalida, "new.json");

      JSONObject json2 = loadJSONObject ("new.json");
      JSONArray values = json2.getJSONArray ("partes");


      for (int i = 0; i < values.size(); i++) {

         JSONObject partes = values.getJSONObject(i);

         String joint = partes.getString("joint");
         float aceleracion = partes.getFloat("aceleracion");
         float velocidad = partes.getFloat("velocidad");

         jointArray[i] = joint;
         aceleracionArray[i] = aceleracion;
         velocidadArray[i] = velocidad;


    println(joint + ", " + aceleracion + ", " + velocidad);
  }
```

Código 1: Paso de datos de *JSON* a *csv*.

```
for (int i=0; i<values.size(); i++){

        TableRow newRow = table.addRow();
        newRow.setString("JOINT", jointArray[i]);
        newRow.setFloat("VELOCIDAD", velocidadArray[i]);
        newRow.setFloat("ACELERACIÓN", aceleracionArray[i]);
        newRow.setFloat("SEGUNDOS",
                        clock.endTime(startTime)/1000)

  }

  saveTable(table, "data/final_taza10.csv");

    }
    while (!stop);
  }
```

Código 2: Paso de datos de *JSON* a *csv* II.

Para alimentar la red se necesitaban características fijas de cada movimiento por ello se decidió hacer la media de cada articulación en cada uno de los archivos, para ello se implementa "*Media.pde*"**.** Este programa carga un archivo *csv*, realiza la operación de media aritmética sobre los datos y el resultado lo guarda en un archivo *csv*.

```
 Table table,tabla_medias;

 void setup(){


table=loadTable("data/valoresfinalesmovsimples/
      final_taza10.csv", "header");

int largo = table.getRowCount();
int num_sets= (largo/15);
println( largo+ "total rows in table");

float [] array_velocidad = new float [15];
float [] array_aceleracion = new float [15];
String [] array_joint = new String [15];

int h=0;

for (int i=0; i<num_sets; i++) {
  for (int j=0; j<15; j++) {
    TableRow row=table.getRow(h);
    array_velocidad [j] += row.getFloat("VELOCIDAD");
    array_aceleracion [j] += row.getFloat("ACELERACIÓN");
    array_joint [j] = row.getString("JOINT");
    h=h+1;}}
```

Código 3: Media.pde.

```
for (int i=0; i<15; i++) {
   array_velocidad [i] = (array_velocidad [i]/num_sets);
   array_aceleracion [i] = (array_aceleracion [i]/num_sets);


    println(array_velocidad[i]+ "array_velocidad [i][j]");
  }


tabla_medias = new Table();

      tabla_medias.addColumn("JOINT");
      tabla_medias.addColumn("VELOCIDAD", Table.FLOAT);
      tabla_medias.addColumn("ACELERACIÓN", Table.FLOAT);



      for (int i=0; i<15; i++){

       TableRow newRow = table.addRow();
       newRow.setString("JOINT", array_joint[i]);
       newRow.setFloat("VELOCIDAD", array_velocidad[i]);
       newRow.setFloat("ACELERACIÓN", array_aceleracion[i]);
  }

saveTable(table,"data/valoresfinalesmovsimples/medias.csv");

}
```

Código 4: Media.pde II.

El programa para entrenar a la red *Red_entrenamiento.pde* consta de los siguientes bloques principales:

- Carga de archivo *csv* con los valores para realizar el entrenamiento.
- Normalización de estos valores.
- Bucle de entrenamiento.
- Cálculo de función de coste.
- Función de *EarlyStopping*

Una vez que la red es entrenada con resultados satisfactorios, todas sus características se guardan en un archivo de extensión *.eg*[12], a su vez este mismo archivo se carga en *Red.pde*. Con este programa se prueba su funcionamiento con entradas que no conoce. A continuación se pasa a describir más detalladamente el proceso de entrenamiento.

[12]Representación XML de una red neuronal de la librería Encog.

Carga de archivos CSV

En esta parte se declara la clase *Table*, en la cual se va a cargar el archivo *csv* que contiene los datos que vamos a utilizar. Los datos se almacenan en 4 arreglos (*arrays*) mediante bucles *for*.

```
String [] jointArray= new String [m*15];
double [] aceleracionArray = new double [m*15];
double [] velocidadArray = new double [m*15];
double [] outputsArray= new double [m*8];
```

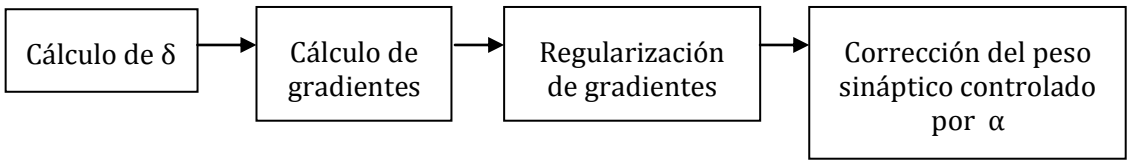
Código 5: Declaración de arreglos de las variables a procesar por la red.

Normalización

El proceso de normalización de las entradas viene definido por la librería Encog; se repite dos veces: una para los valores de velocidad y otra para los valores de aceleración. El rango de normalización es entre 0 y 1. (Heaton J. , 2010)

Red

El algoritmo principal de la red trabaja con conjuntos entrada-salida cumpliendo el siguiente esquema:



La estructura de la red es la siguiente:

```
FeedForwardPattern pattern = new FeedForwardPattern();
pattern.setActivationFunction(new ActivationSigmoid());
pattern.setInputNeurons(30);
pattern.addHiddenLayer(20);
pattern.addHiddenLayer(20);
pattern.setOutputNeurons(8);
pattern.generate();
```

Código 6: Declaración de red Feedforward de dos capas ocultas.

Consta de una capa de entrada, una capa de salida y dos capas ocultas. El número de neuronas en cada una de las capas se puede modificar simplemente sustituyendo los valores numéricos. El patrón establecido para la red es *Feed Forward* ó hacia delante.

A continuación se establecen los pesos iniciales, que están limitados entre 1 y -1.

```
new RangeRandomizer(-1,1).randomize(network);
```

Código 7: Declaración de pesos sinápticos aleatorios.

El bucle principal de la red es un bucle *do-while*, es decir, el entrenamiento se ejecutará hasta llegar al número de iteraciones propuesto o alcanzar un error máximo. Para comenzar el cálculo se guardan una pareja de valores de las matrices de entrada y salida en dos arrays, inputs[] y outputs[], y se trabajará con estos a partir de ahora. Una iteración se considera completa cuando han pasado por la red todos los valores de entrada-salida del set de entrenamiento.

Esquema de una iteración completa para un grupo $m = 3$.

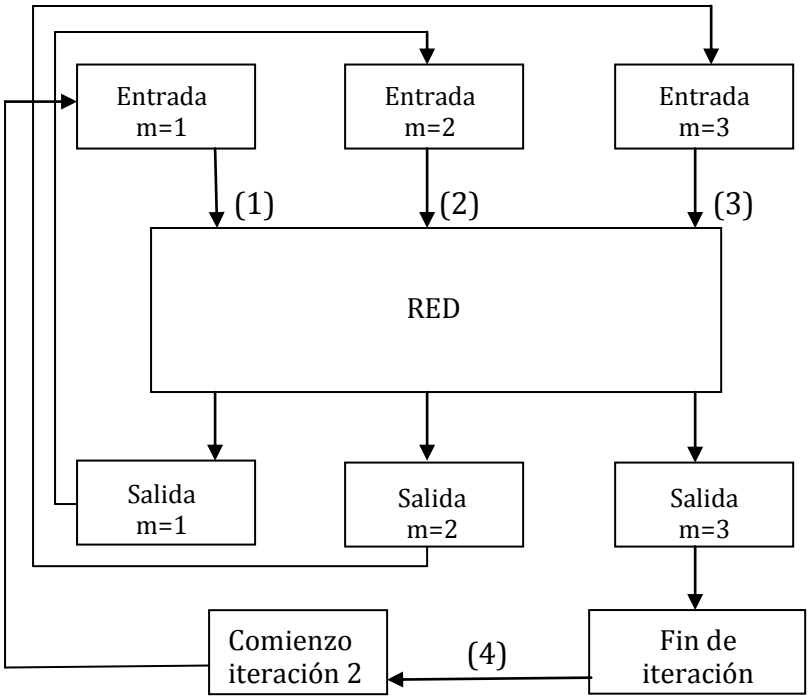


Figura 22: Proceso de iteración.

Como puede verse las entradas pasan un total de 3 veces por el procesado de la red, una vez por cada entrada, pero sólo cuenta como 1 iteración.

Se comienza con el cálculo de las δ de la capa de salida y las dos capas ocultas.

δ Capa de salida:

Se restan la salida actual y la salida esperada para esa entrada, como en (13).

```
deltas_outputlayer[i]=actual_output[i]-outputs[i];
```

Código 8: Cálculo deltas capa de salida.

δ Capa oculta 2:

Para calcular el error de una capa oculta se utiliza (14). Se declara la variable sum_weights[] que almacena los resultados parciales de multiplicar los pesos que conectan cada neurona de la capa oculta 2 con la capa de salida, y por la delta de la neurona con la que conecta.

```
double sum_weights[]=new double[toNeuronCount2];

int var=0;
for(int i=0; i<toNeuronCount3; i++){
          for(int j=0; j<toNeuronCount2; j++){

    sum_weights[j]+=weights[var]*deltas_outputlayer[i];

        deltas_hiddenlayer2[j]=sum_weights[j]*(
        output_neurons_hiddenlayer2[j]*
        (1-output_neurons_hiddenlayer2[j]));

           var=var+1;
           }
 var=var+1;
 }
```

Código 9: Cálculo deltas en la segunda capa oculta.

δ Capa oculta 1:

El proceso es igual que el anterior.

```
double []sum_weights2=new double[toNeuronCount1];

   for(int i=0; i<toNeuronCount2; i++){
       for(int j=0; j<toNeuronCount1; j++){


   sum_weights2[j]+=weights[var]*deltas_hiddenlayer2[i];
         deltas_hiddenlayer1[j]=sum_weights2[j]*
         (output_neurons_hiddenlayer1[j]*
         (1-output_neurons_hiddenlayer1[j]));

    var=var+1;
    }
 var=var+1;
 }
}
```

Código 10: Cálculo deltas en la primera capa oculta.

Una vez se tienen los valores de las δ se pasa a calcular los gradientes.

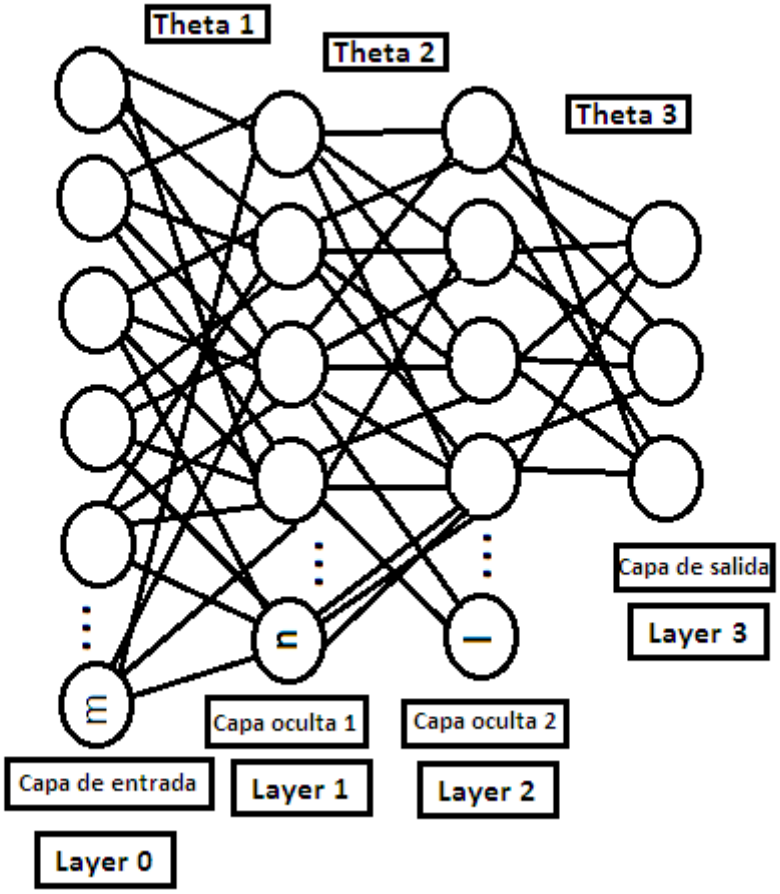


Figura 23: Red neuronal indicando nombre de las capas.

Cálculo de gradiente y regularización entre capa de entrada y capa oculta1:

Para el cálculo de los gradientes se utilizan las ecuaciones (15), (16) y (17), dependiendo de si el peso conecta una neurona *bias* o no.

```
int var1=0;
      for (inti=0; i<toNeuronCount1;i++){
        for (int j=0; j<(toNeuronCount0+1);j++){

    Theta1_grad[var1]=Theta1_grad[var1]+
    (inputs_theta[j]*deltas_hiddenlayer1[i]);

        var1=var1+1;
            }
          }
```

Código 11: Cálculo de gradiente y regularización entre capa de entrada y capa oculta 1.

```
int x=0;

for(inti=0; i<(toNeuronCount0+1)*toNeuronCount1; i++){

   if(toNeuronCount0+(toNeuronCount0+1)*x ==i){

       Theta1_grad[i]=(Theta1_grad[i]/m);
       x=x+1;
       }

    else
    {
    Theta1_grad[i]=(Theta1_grad[i]/m)+
    Theta1_grad[i]*(lambda/m);
    }
  }
```

Código 12: Cálculo de gradiente y regularización entre capa de entrada y capa oculta 1 II.

Cálculo de gradiente y regularización entre capa oculta1 y capa oculta 2:

Al igual que en el caso anterior se utilizan las ecuaciones (15), (16) y (17), dependiendo de si el peso conecta una neurona *bias* o no.

```
int var2=0;
        for (inti=0; i<toNeuronCount2 ;i++){
          for (int j=0; j<toNeuronCount1+1 ;j++){

    Theta2_grad[var2]=Theta2_grad[var2]+
    (output_neurons_hiddenlayer1[j]
    *deltas_hiddenlayer2[i]);

   var2=var2+1;
          }
      }

int y=0;
for(int i=0; i<(toNeuronCount1+1)*toNeuronCount2; i++){

       if(toNeuronCount1+(toNeuronCount1+1)*y ==i){

         Theta2_grad[i]=(Theta2_grad[i]/m);
         y=y+1;
 }
 else{

    Theta2_grad[i]=(Theta2_grad[i]/m)+Theta2_grad[i]*
    (lambda/m);
      }
  }
```

Código 13: Cálculo de gradiente y regularización entre capa oculta1 y capa oculta 2.

Cálculo de gradiente y regularización entre capa oculta 2 y capa de salida:

En este caso también se siguen las mismas pautas que en los dos casos anteriores.

```
int var3=0;

   for (inti=0; i<toNeuronCount3 ;i++){
     for (int j=0; j<toNeuronCount2+1 ;j++){

    Theta3_grad[var3]=Theta3_grad[var3]+
    (output_neurons_hiddenlayer2[j]*
    deltas_outputlayer[i]);

    var3=var3+1;
            }
        }

int z=0;

for(inti=0; i<(toNeuronCount2+1)*toNeuronCount3; i++){

    if(toNeuronCount2+(toNeuronCount2+1)*z ==i){
       Theta3_grad[i]=(Theta3_grad[i]/m);
       z=z+1;
       }
       else{

Theta3_grad[i]=(Theta3_grad[i]/m)+(Theta3_grad[i]
*(lambda/m));
}
 }
```

Código 14: Cálculo de gradiente y regularización entre capa oculta 2 y capa de salida.

Por último se aplica (17) y se reemplazan todos los pesos de la red.

```
weights[i]=weights[i]-alfa*THETA[i];
```

Código 15: Sustitución de los nuevos pesos sinápticos.

Función de coste

Para el cálculo de esta función descrita en (12) se pasan a la función los siguientes valores: λ, α, m (número de conjuntos entrada-salida), el array que contiene las salidas deseadas, el array de pesos y el array que contiene las salidas reales para cada valor de m dentro de la misma iteración.

```
static double cost_function(double[] output_cost_function,
   double[][] OUTPUTS, int m, double lambda, double[] weights) {

          double costfunction_term = 0;
          double sum_weights_square = 0;

         int h=0;

          for(inti=0; i<m ;i++){
         for(int j=0; j<8; j++){

costfunction_term += (OUTPUTS[i][j])*Math.log10((float) output_cost_function[h])
        +(1-OUTPUTS[i][j])*Math.log10((float) (1-output_cost_function[h]));

 h=h+1;
   }
 }

    costfunction_term= -(lambda/m)*costfunction_term;
    System.out.println( "costfunction_term = " + costfunction_term);


     for (int j=0; j<weights.length;j++){
    sum_weights_square = (weights[j])*(weights[j]);}


       double regularization= (lambda/m)*sum_weights_square;
    System.out.println( "regularizacion = " + regularization);


       double value = costfunction_term + regularization;

    returnvalue;
 }
```

Código 16: Función de coste.

Early Stopping

Se declara como variable booleana, cuando *Early Stopping* es verdadero, el entrenamiento se parará en cuanto la función de coste supere el valor de su último resultado, igualando el número de iteraciones a las definidas al principio del bucle.

```
  if (earlyStopping == true){

         if(iteration>m
&&array_cost_function[p]>array_cost_function[p-1]){

  System.out.println( "numero_iteraciones"+iteration_count);
  System.out.println( "iteracion"+iteration);
  System.out.println( "Early stopping");
  iteration=iteration_count;
          }
      }
  p=p+1;
```

Código 17: Función de *Early Stopping*.

En cuanto a "*Red.pde",* la gran diferencia con "*Red_entrenamiento.pde*" es la ausencia del bucle *do-while*. En este caso se cargan los datos de la red ya entrenada y el arreglo de los pesos finales.

```
finalStringFILENAME="prueba_final_valores_combinados11.eg";

    System.out.println("Loading network");
    BasicNetworknetwork=(BasicNetwork)
        EncogDirectoryPersistence.loadObject
        (new File(FILENAME));

    double [] pesos= NetworkCODEC.networkToArray(network);
```

Código 18: Proceso de carga de datos de la red neuronal.

El resto del proceso es el que se sigue en "*Red_entrenamiento.pde*". Se cargan archivos *csv* con las variables, se normalizan los valores y se comprueban las salidas y el error de la red para esos valores.

4.1.1 Entrenamiento de la red para movimientos simples

Se entrena la red con los 8 movimientos descritos en el capítulo 3. Se comienza con un número bajo de neuronas en las capas ocultas, que se irá modificando en función de los resultados, se sigue el mismo proceso para α, λ y número de iteraciones.

Se han grabado 5 vídeos por movimiento, por lo que como se ha explicado anteriormente se dividen estos datos en conjunto de entrenamiento (60%), conjunto de validación cruzada (20%) y conjunto de prueba (20%).

En el Anexo A se pueden consultar los resultados exactos de la red para cada movimiento; se señalan las neuronas ganadoras para cada caso.

Prueba 1

	Neuronas capa oculta 1	Neuronas capa oculta 2	α	λ	Nº Iteraciones	J(Θ)	Tiempo (s)	Acierto (%)
Entrenamiento	10	10	0.07	0.01	15000	0.013	5.946	-
VC	10	10	0.07	0.01	15000	0.009	5.789	-
Prueba	10	10	-	0.01	-	0.013	0.236	25

Tabla 6: Resultado y configuración de la red para la primera prueba de movimientos simples.

Se comienza con un número bajo de neuronas, λ y α. En este caso se observa un error bajísimo de ambos sets, sin embargo, los resultados son bastantes pobres. Sólo acierta un 25% de los movimientos.

Prueba 2

	Neuronas capa oculta 1	Neuronas capa oculta 2	α	λ	Nº Iteraciones	J(Θ)	Tiempo (s)	Acierto (%)
Entrenamiento	15	10	0.07	0.01	15000	0.054	7.306	-
VC	15	10	0.07	0.01	15000	0.017	7.187	-
Prueba	15	10	-	0.01	-	0.026	0.255	25

Tabla 7: Resultado y configuración de la red para la segunda prueba de movimientos simples.

Se aumenta el número de neuronas por capa. De nuevo un 25% de aciertos, diferentes al conjunto anterior. Los dos errores son muy bajos, pero se aprecia la diferencia entre el set de entrenamiento y el de validación cruzada, esa diferencia indica un problema de *high variance*, para intentar solucionarlo se aumenta λ.

Prueba 3

	Neuronas capa oculta 1	Neuronas capa oculta 2	α	λ	Nº Iteraciones	J(Θ)	Tiempo (s)	Acierto (%)
Entrenamiento	15	10	0.07	0.08	15000	0.638	7.392	-
VC	15	10	0.07	0.08	15000	0.109	7.009	-
Prueba	15	10	-	0.08	-	0.034	0.251	25

Tabla 8: Resultado y configuración de la red para la tercera prueba de movimientos simples.

Aumenta el error y se incrementa la diferencia entre los dos sets.

Prueba 4

	Neuronas capa oculta 1	Neuronas capa oculta 2	α	λ	Nº Iteraciones	J(Θ)	Tiempo (s)	Acierto (%)
Entrenamiento	15	10	0.07	0.32	15000	0.407	6.875	-
VC	15	10	0.07	0.32	15000	0.702	7.162	-
Prueba	15	10	-	0.32	-	0.364	0.23	25

Tabla 9: Resultado y configuración de la red para la cuarta prueba de movimientos simples.

Se incrementa de nuevo λ pero aumenta el error de nuevo sin grandes resultados, tampoco se corrige el problema de *high variance*.

Prueba 5

	Neuronas capa oculta 1	Neuronas capa oculta 2	α	λ	Nº Iteraciones	J(Θ)	Tiempo (s)	Acierto (%)
Entrenamiento	15	15	0.5	0.01	15000	0.053	9.055	-
VC	15	15	0.5	0.01	15000	4.235	8.32	-
Prueba	15	15	-	0.01	-	0.218	0.304	87.5

Tabla 10: Resultado y configuración de la red para la quinta prueba de movimientos simples.

En esta prueba se aumenta el número de neuronas, se disminuye el valor de λ y aumenta α para que aprenda más rápido. Como se puede comprobar los resultados mejoran notablemente con un 87,5% de aciertos.

Prueba 6

	Neuronas capa oculta 1	Neuronas capa oculta 2	α	λ	Nº Iteraciones	J(Θ)	Tiempo (s)	Acierto (%)
Entrenamiento	20	15	0.5	0.01	15000	0.059	9.683	-
VC	20	15	0.5	0.01	15000	0.004	9.598	-
Prueba	20	15	-	0.01	-	0.024	0.238	87.5

Tabla 11: Resultado y configuración de la red para la sexta prueba de movimientos simples.

Si se continúa aumentando el número de neuronas se vuelve a conseguir un resultado muy bueno, 87,5% de aciertos y los errores son muy bajos. De nuevo es el movimiento 3 el que no reconoce.

Prueba 7

	Neuronas capa oculta 1	Neuronas capa oculta 2	α	λ	Nº Iteraciones	J(Θ)	Tiempo (s)	Acierto (%)
Entrenamiento	20	20	0.5	0.01	15000	0.029	10.895	-
VC	20	20	0.5	0.01	15000	0.066	10.733	-
Prueba	20	20	-	0.01	-	0.014	0.238	75

Tabla 12: Resultado y configuración de la red para la séptima prueba de movimientos simples.

Se aumenta de nuevo el número de neuronas, ahora hay 20 en cada capa oculta. El resultado de aciertos disminuye, aunque el error sigue siendo muy bajo, y se puede ver como aumenta el tiempo de procesado cada vez que sube el número de neuronas que participan en la operación.

Prueba 8

	Neuronas capa oculta 1	Neuronas capa oculta 2	α	λ	Nº Iteraciones	J(Θ)	Tiempo (s)	Acierto (%)
Entrenamiento	20	20	0.5	0.32	20000	0.095	14.269	-
VC	20	20	0.5	0.32	20000	0.794	15.386	-
Prueba	20	20	-	0.32	-	0.134	0.249	87.5

Tabla 13: Resultado y configuración de la red para la octava prueba de movimientos simples.

Para esta prueba se aumenta λ y el número de iteraciones, con esto se comprueba cómo afecta en su aprendizaje. Al aumentar estos dos parámetros y teniendo en cuenta los últimos resultados, el desempeño de la red debe mejorar, obteniéndose un 87,5% de aciertos.

Prueba 9

	Neuronas capa oculta 1	Neuronas capa oculta 2	A	λ	Nº Iteraciones	J(Θ)	Tiempo (s)	Acierto (%)
Entrenamiento	25	20	0.5	0.32	20000	0.344	17.118	-
VC	25	20	0.5	0.32	20000	0.005	16.735	-
Prueba	25	20	-	0.32	-	0.2	0.241	100

Tabla 14: Resultado y configuración de la red para la novena prueba de movimientos simples.

Se aumenta de nuevo el número de neuronas. Para esta combinación acierta todas las entradas, tenemos una red que reconoce todos los movimientos para los que ha sido entrenada. Además los dos errores son muy bajos y con tendencia a cero.

Prueba 10

	Neuronas capa oculta 1	Neuronas capa oculta 2	α	λ	Nº Iteraciones	J(Θ)	Tiempo (s)	Acierto (%)
Entrenamiento	25	25	0.5	0.32	15000	1.227	19.397	-
VC	25	25	0.5	0.32	15000	0.513	24.46	-
Prueba	25	25	-	0.32	-	0.476	0.257	100

Tabla 15: Resultado y configuración de la red para la décima prueba de movimientos simples.

Una última prueba igualando las neuronas de la 2ª capa oculta a las de la primera. De nuevo obtenemos un acierto total de los movimientos, aunque un mayor error y tiempo de procesado.

A continuación se presentan las gráficas de las curvas de aprendizaje de las pruebas anteriores, en el eje de abcisas se representa el número de iteraciones y en el de ordenadas el valor del error:

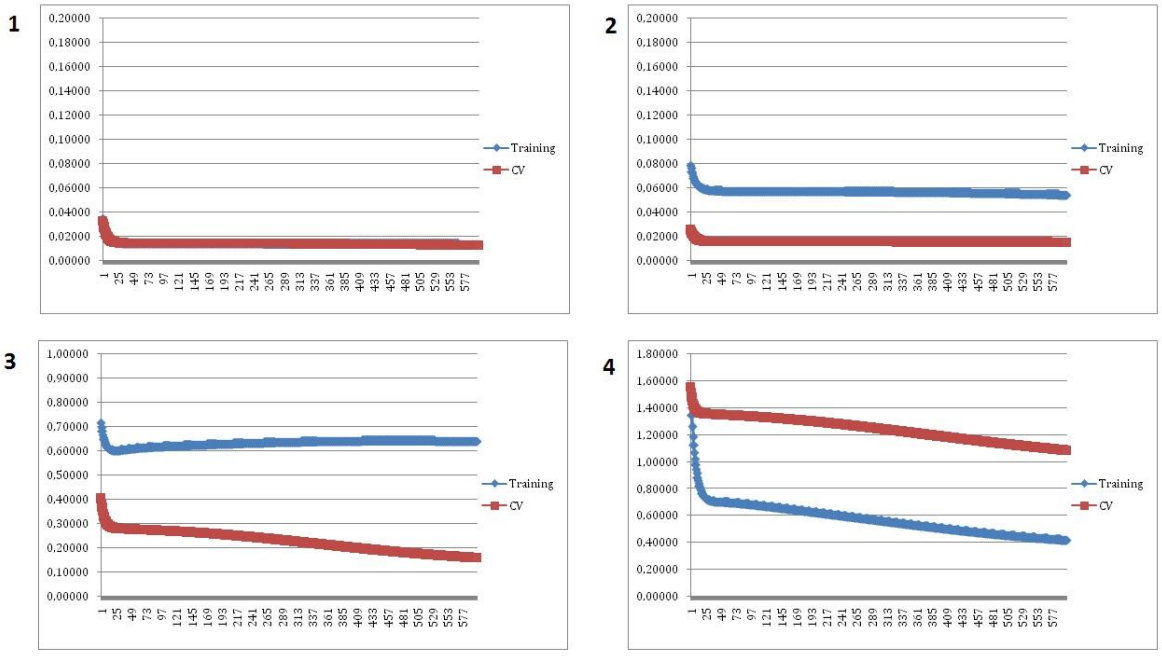


Figura 24: Curvas de aprendizaje para prueba con movimientos simples 1 - 4.

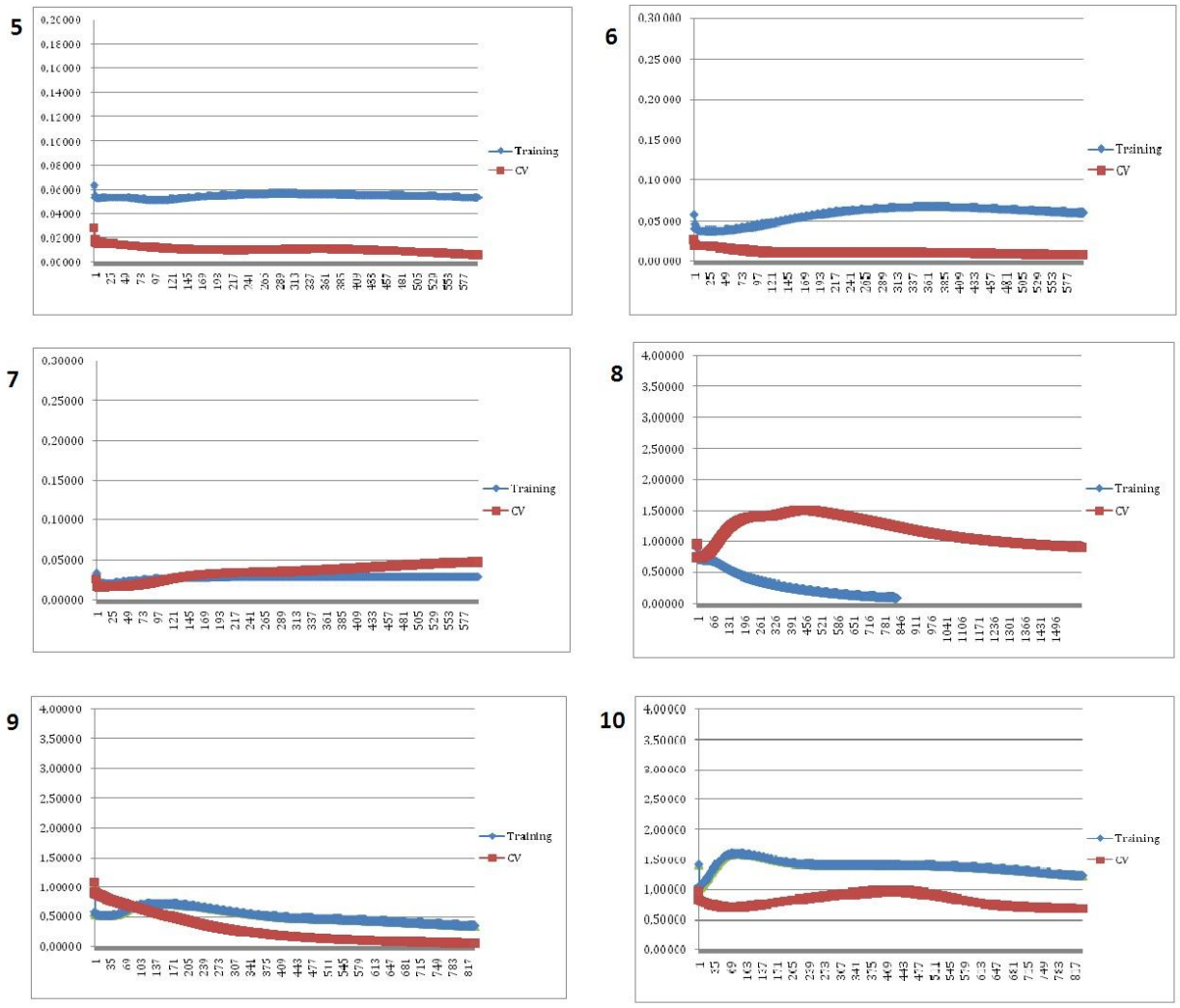


Figura 25: Curvas de aprendizaje para prueba con movimientos simples 5 - 10.

4.2.1 Entrenamiento de la red para movimientos complejos

Prueba 1

	Neuronas capa oculta 1	Neuronas capa oculta 2	α	λ	Nº Iteraciones	J(Θ)	Tiempo (s)	Acierto (%)
Entrenamiento	10	10	0.07	0.05	15000	0.032	6.003	-
VC	10	10	0.07	0.05	15000	0.034	6.276	-
Prueba	10	10	-	0.05	-	1.248	0.352	50

Tabla 16: Resultado y configuración de la red para la primera prueba de movimientos complejos.

Como puede observarse en las curvas los dos errores son bajos, pero la red no termina de acertar todos los resultados. Por lo que se prueba subiendo el número de neuronas de las capas ocultas.

Prueba 2

	Neuronas capa oculta 1	**Neuronas capa oculta 2**	**α**	**λ**	**Nº Iteraciones**	**J(Θ)**	**Tiempo (s)**	**Acierto (%)**
Entrenamiento	15	10	0.07	0.05	15000	0.186	7.507	-
VC	15	10	0.07	0.05	15000	0.128	9.142	-
Prueba	15	10	-	0.05	-	1.236		75

Tabla 17: Resultado y configuración de la red para la segunda prueba de movimientos complejos.

En este caso se ve un claro ejemplo de *high variance*, hay una gran diferencia entre los dos errores, a pesar de ello tiene un 75% de aciertos. Se puede observar como aumenta el tiempo de procesado con el aumento de las neuronas por capa. Se trata de arreglar el problema de *high variance* incrementando λ.

Prueba 3

	Neuronas capa oculta 1	**Neuronas capa oculta 2**	**α**	**λ**	**Nº Iteraciones**	**J(Θ)**	**Tiempo (s)**	**Acierto (%)**
Entrenamiento	15	10	0.07	0.1	15000	3.350	7.666	-
VC	15	10	0.07	0.1	15000	0.070	8.214	-
Prueba	15	10	-	0.1	-	1.457		50

Tabla 18: Resultado y configuración de la red para la tercera prueba de movimientos complejos.

Se ha aumentado λ para evitar un error que se ha pronunciado mucho más. Este problema puede deberse a los pesos iniciales. Se repite la operación con distintos pesos iniciales.

Prueba 4

	Neuronas capa oculta 1	**Neuronas capa oculta 2**	**α**	**λ**	**Nº Iteraciones**	**J(Θ)**	**Tiempo (s)**	**Acierto (%)**
Entrenamiento	15	10	0.07	0.1	15000	0.064	7.718	-
VC	15	10	0.07	0.1	15000	0.149	8.481	-
Prueba	15	10	-	0.1	-	0.107	0.266	75

Tabla 19: Resultado y configuración de la red para la cuarta prueba de movimientos complejos.

De nuevo un 75% de aciertos. Hay un movimiento que no reconoce en ninguno de los dos casos que se ha obtenido este resultado, a continuación se prueba aumentando el número de neuronas por capa.

Prueba 5

	Neuronas capa oculta 1	**Neuronas capa oculta 2**	**α**	**λ**	**Nº Iteraciones**	**J(Θ)**	**Tiempo (s)**	**Acierto (%)**
Entrenamiento	15	15	0.07	0.1	15000	2.719	8.547	-
VC	15	15	0.07	0.1	15000	0.228	8.860	-
Prueba	15	15	-	0.1	-	0.741	0.336	50

Tabla 20: Resultado y configuración de la red para la quinta prueba de movimientos complejos.

Se aprecia una diferencia en el error representado en las curvas, se aumenta λ para tratar de reducir esa diferencia.

Prueba 6

	Neuronas capa oculta 1	**Neuronas capa oculta 2**	**α**	**λ**	**Nº Iteraciones**	**J(Θ)**	**Tiempo (s)**	**Acierto (%)**
Entrenamiento	15	15	0.07	0.32	15000	0.050	8.314	-
VC	15	15	0.07	0.32	15000	1.025	9.406	-
Prueba	15	15	-	0.32	-	0.450	0.313	75

Tabla 21: Resultado y configuración de la red para la sexta prueba de movimientos complejos.

El error de entrenamiento baja pero el de CV se mantiene constante. El porcentaje de aciertos aumenta hasta el 75%. Se prueba aumentando las neuronas por capa oculta.

Prueba 7

	Neuronas capa oculta 1	**Neuronas capa oculta 2**	**α**	**λ**	**Nº Iteraciones**	**J(Θ)**	**Tiempo (s)**	**Acierto (%)**
Entrenamiento	20	15	0.07	0.32	15000	0.931	10.533	-
VC	20	15	0.07	0.32	15000	0.163	10.572	-
Prueba	20	15	-	0.32	-	1.243	0.298	50

Tabla 22: Resultado y configuración de la red para la séptima prueba de movimientos complejos.

Los dos errores son decrecientes pero sin buenos resultados finales, sólo reconoce el 50% de las muestras. Se sigue aumentando el número de neuronas por capa.

Prueba 8

	Neuronas capa oculta 1	Neuronas capa oculta 2	α	λ	Nº Iteraciones	J(Θ)	Tiempo (s)	Acierto (%)
Entrenamiento	20	20	0.1	0.32	15000	1.659	11.265	-
VC	20	20	0.1	0.32	15000	0.427	11.769	-
Prueba	20	20	-	0.32	-	1.161	0.290	75

Tabla 23: Resultado y configuración de la red para la octava prueba de movimientos complejos.

Se ha aumentado α también para acelerar el proceso de aprendizaje. Sigue habiendo un problema de *high variance* que no se ha corregido al aumentar λ.

Prueba 9

	Neuronas capa oculta 1	Neuronas capa oculta 2	α	λ	Nº Iteraciones	J(Θ)	Tiempo (s)	Acierto (%)
Entrenamiento	25	20	0.1	0.64	17000	3.280	14.887	-
VC	25	20	0.1	0.64	17000	0.110	16.407	-
Prueba	25	20	-	0.64	-	1.521	0.303	62.5

Tabla 24: Resultado y configuración de la red para la novena prueba de movimientos complejos.

Se dobla el valor de lambda y aumenta el número de iteraciones y no se corrige el problema de high variance sino que aumenta. El porcentaje de aciertos disminuye.

Prueba 10

	Neuronas capa oculta 1	Neuronas capa oculta 2	α	λ	Nº Iteraciones	J(Θ)	Tiempo (s)	Acierto (%)
Entrenamiento	25	25	0.1	0.64	17000	0.795	16.645	-
VC	25	25	0.1	0.64	17000	0.020	17.239	-
Prueba	25	25	-	0.64	-	1.976	0.297	75

Tabla 25: Resultado y configuración de la red para la décima prueba de movimientos complejos.

Se aumenta de nuevo el número de neuronas por capa oculta consiguiendo un acierto del 75%.

A continuación se presentan las gráficas de las curvas de aprendizaje de las pruebas anteriores, en el eje de abcisas se representa el número de iteraciones y en el de ordenadas el valor del error:

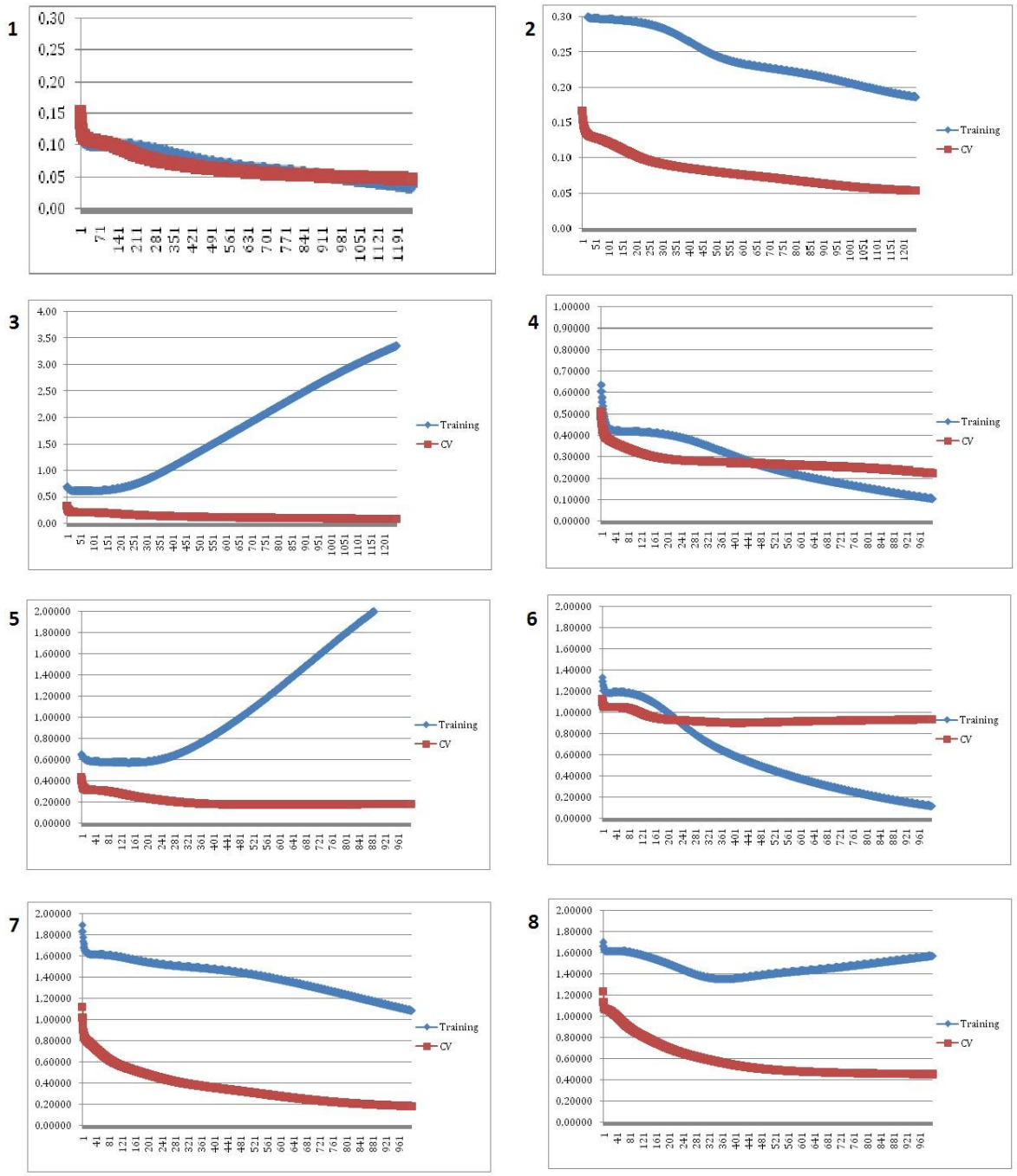


Figura 26: Curvas de aprendizaje para prueba con movimientos complejos 1 - 8.

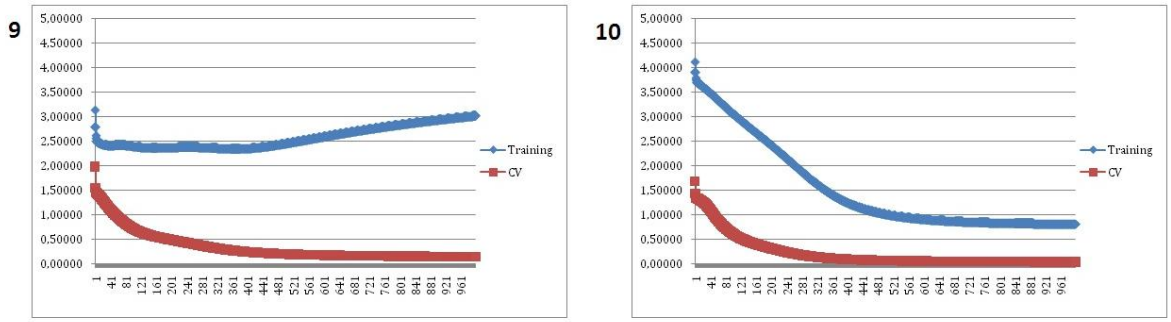


Figura 27: Curvas de aprendizaje para prueba con movimientos complejos 9 - 10.

4.2 Conclusiones

Se ha comprobado la influencia de cada parámetro en el rendimiento de la red durante su entrenamiento y posterior prueba. El aumento del número de neuronas, en pasos de 5 en 5 y siempre manteniendo la estructura de pirámide invertida, ha permitido apreciar como es directamente proporcional al tiempo de procesado aunque tanbién lo son los resultados obtenidos. Los pesos iniciales también juegan un papel muy importante, ya que dependiendo en que punto de la función de coste se encuentren se pueden obtener mejores ó peores resultados. En ambos casos los resultados erróneos pertenecen a movimientos muy parecidos en ejecución por lo cual la red los confunde. La red para movimientos simple obtiene muy buenos resultados, llegando varios de estos a ser del 100%. Sin embargo la red para movimientos complejos obtine como máximo un 75% lo cual no es un mal resultado pero más bajo que el obtenido para la anterior.

Capítulo 5

Conclusiones y líneas futuras

5.1 Conclusiones finales

El objetivo que se perseguía con este proyecto era el de la identificación de un movimiento a partir de valores cinemáticos registrados durante la ejecución del mismo. Para su consecución se necesitaban conocimientos en dos áreas concretas, estudio del movimiento corporal y redes neuronales artificiales.

La forma de moverse de una persona está relacionada con su estado anímico así como con los rasgos de la personalidad del individuo. Se ha comprobado como a pesar de ser un método de evaluación sobre todo utilizado en danza y arte dramático es muy útil en otros campos. Se han presentado las 4 categorías en las que se divide el esfuerzo y las polaridades de cada una de estas. Para caracterizar a cada uno de los elementos de esfuerzo se han buscado movimientos en los que predominase esta característica, de ahí que el número de movimientos simples sea 8. El caso de los movimientos combinados trata de dar un paso más en la identificación de estos valores, combinándolos por parejas se pretende que el sistema implementado sea capaz de reconocer varias características a la vez ya que en el mundo real los movimientos generalmente están compuestos por varios de estos elementos.

En lo referente a las redes neuronales se ha comprobado que son una herramienta de trabajo potente y eficaz. Tanto la elección del tipo de red como su configuración dependen de variables que se han de seleccionar según las características de cada caso concreto.

El objetivo de este proyecto se ha cumplido con resultados satisfactorios. Se partía de un programa con el que se obtenían valores cinemáticos de las articulaciones, al que se han incorporado varios cambios en el código para la obtención de los datos en un formato que pudiera manejar la red y la grabación en vídeo de los movimientos realizados. Se ha programado una red neuronal estática de dos capas ocultas capaz de reconocer los 8 movimientos para los que ha sido entrenada. Durante el proceso de desarrollo se han tomado decisiones que han afectado positivamente al resultado obtenido. Un cambio clave ha sido la manera de presentación de los datos a la red. En un principio se presentaban todos los datos recogidos de cada movimiento durante un intervalo de tiempo determinado y se alimentaba la red directamente. La red al ser estática no era capaz de diferenciar características de cambio temporales por lo que se decidió hacer una media de los valores de aceleración y velocidad de cada articulación. El resultado ha sido mejor de lo que inicialmente se esperaba ya que para dos de las diez configuraciones que se presentan la red ha otenido un 100% de aciertos. La variación en parámetros como la tasa de aprendizaje, α, ó el término de regularización, λ, también

influye, se ha comprobado como a partir de ciertos valores la red incrementa su tiempo de procesado pero que en algunos casos no tiene importancia debido a la buena función que realiza.

La prueba con movimientos complejos, combinación de dos elementos de esfuerzo, ha tenido un rendimiento menor. La red ha confundido en varias ocasiones movimientos parecidos. Para este caso creo que habría que extraer más características que relacionasen los movimientos.

Si comparamos el desarrollo de este proyecto con el único de los expuestos al principio de este documento que también utilizaba redes neuronales, se puede afirmar que consigue mejores resultados con una sola red neuronal que aquel en el que se programaban 4 redes, una por cada factor de esfuerzo, sin embargo, este último con sus 4 redes era capaz de identificar movimientos complejos. Otro punto positivo es su desarrollo con *software* libre, tanto la plataforma Processing como las librerías utilizadas son de fácil acceso para todo aquel que esté interesado.

El hecho de haber logrado cuantizar algo que se realiza de manera subjetiva por expertos en el tema es un gran avance, y dado los buenos resultados obtenidos este trabajo sirve de comienzo a un estudio más avanzado de la cuestión.

5.2 Líneas futuras de trabajo

5.2.1 Redes dinámicas

Se propone como mejora abordar el mismo problema pero con una red dinámica o también conocida como red de Elman. Este tipo de redes cuenta con lazos de retroalimentación lo que les permite reconocer secuencias. Esto último permitiría analizar el movimiento en tiempo real, capturando las entradas con kinect y alimentándolas directamente a la red. De esta manera también se prescindiría de programas intermedios haciendo más rápido su proceso.

Una aplicación así permitiría el estudio de la persona *in-situ* facilitando la labor del observador.

5.2.2 Triangulación

Otra aportación interesante sería la captación de datos con más de una cámara. Con este sistema podrían extraerse más datos característicos de cada movimiento y facilitar su identificación por la red, como sucedía en el caso de los movimientos complejos. Con una red de dos ó incluso tres cámaras kinect aumentamos el rango de visibilidad de la acción y por tanto el volumen de información que recibimos de la escena.

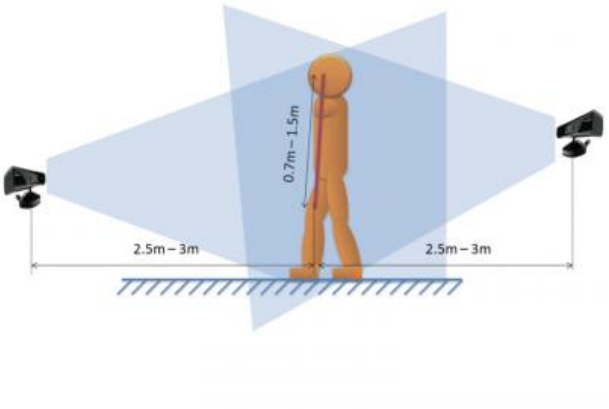


Figura 28: Ejemplo de triangulación con kinect[13]

5.2.3 Acelerómetros

En lugar de utilizar un kinect para recoger los datos se podría cambiar por un acelerómetro. La ventaja de estos dispositivos es que no dependen del ángulo con el que enfoquen al sujeto, además de encontrarse en multitud de dispositivos hoy en día como pueden ser un teléfono móvil o el mando de la videoconsola Wii. En el caso del teléfono móvil se puede tener acceso además a datos de GPS, vídeo, audio… abriendo un camino de nuevas posibilidades.

[13] http://wiki.ipisoft.com/User_Guide_for_Dual_Depth_Sensor_Configuration

PRESUPUESTO

Material utilizado

	Unidades	Precio	Total
Ordenador sobremesa	1	300	300
Cámara kinect	1	59	59

Tabla 26: Presupuesto material utilizado.

TOTAL: 359 €

Horas de trabajo

	Horas	Precio/Hora	Total
Estudios previos	60	25	1500
Diseño y programación	300	25	7500
Pruebas	160	25	4000

Tabla 27: Presupuesto horas de trabajo.

TOTAL: 13.000 €

Coste total del proyecto: 13.359 €

BIBLIOGRAFÍA

Birdwhistell, R. L. (1952). *Introduction to kinesics: an annotation system for analysis of body motion and gesture.* Kentucky: University of Louisville.

Blanco Vega, M. J. (2009). *Enfoques teóricos sobre la expresión corporal como medio de formación y comunicación.* Iberoamericana Institución Universitaria.

Castañer, M. (2000). *Expresión corporal y danza.* Barcelona: INDE.

Chóliz, M. (1995). *La expresión de las emociones en la obra de Darwin.* Valencia: Prácticas de historia de la psicología. Promolibro.

Connett, D. M. (2011, 8 1). *Total Body Integration: A Phenomenological Heuristic Study.* Retrieved 2013, from http://digitalcommons.colum.edu/cgi/viewcontent.cgi?article=1018&context=theses_dmt

Davis, F. (2010). *La comunicación no verbal.* Alianza editorial.

Gestal, M. (n.d.). *S.A.B.I.A.* Retrieved 2014, from http://sabia.tic.udc.es/mgestal/cv/RNAtutorial/TutorialRNA.pdf

Hall, E. T. (1959). *The Silent Language.* Anchor books.

Hall, J. C. (2011, 12 22). *How to Do Gesture Recognition With Kinect Using Hidden Markov Models (HMMs)*. Retrieved 2014, from http://www.creativedistraction.com/demos/gesture-recognition-kinect-with-hidden-markov-models-hmms/

Haykin, S. (1999). *Neural Networks. A comprehensive Foundation.* New Jersey: Prentice Hall International.

Heaton, J. (2005). *Programming Neural Networks in Java.* St. Louis: Heaton Research.

Heaton, J. (2010). *Programming Neural Networks with Encog 2 in Java.* St. Louis: Heaton Research.

Hosmer, D. W., & Lemeshow, S. (2000). *Applied Logistic Regression (2nd edition).* Wiley.

Jiahui, W., Gang, P., Daqing, Z., Guande, Q., & Shijian, L. (2013). *Gesture Recognition with a 3-D Accelerometer.* Hangzhou: Department of Computer Science, Zhejiang University.

Jui-Fa, C., Wei-Chuan, L., Kun-Hsiao, T., & Shih-Yao, D. (2011). Analysis and Evaluation of Human Movement based on Laban Movement Analysis. *Tamkang Journal of Science and Engineering* .

Konie, R. (2011). *movementhasmeaning.com.* Retrieved 2014, from http://movementhasmeaning.com/wp-content/uploads/2010/09/LMA-Workshop-Sheet.pdf

Levy, F. (1988). *The evolution of modern dance therapy.* Journal of Physical Education,Recreation & Dance.

Newlove, J., & Dalby, J. (2004). *Laban for all.* London: Nick Hern Books Limited.

Ng, P. A. (2011). *Coursera.* Retrieved 2013, from ml-class.org

Ramos, D. (2013). *Estudio cinemático del cuerpo humano mediante kinect.* Madrid: Universidad Politécnica de Madrid.

Ruesch, J., & Kees, W. (1972). *Nonverbal Communication: Notes on the Visual Perception of Human Relations.* California: University of California Press.

Schinca, M. (2002). *Expresión corporal: Técnicas y expresión del movimiento (4ª ed.).* Wolters Kluwer Educacion.

Trinity Laban Conservatoire of Music and Dance. (n.d.). Retrieved from http://www.trinitylaban.ac.uk/about-us/our-history/rudolf-laban

Zao, L. (2001). *Synthesis and acquisition of Laban movement.* Pennsylvania: University of Pennsylvania.

ANEXO A: Instalación de librerías y hardware

Instalación de Processing

Este entorno Java creado por Casey Reas y Ben Fry, fue concebido principalmente para enseñar programación con un trasfondo en las artes visuales, de ahí que a sus programas se los denominen bocetos (*sketch*). Para descargar este programa habrá que ir a su sitio web www.processing.org, y en la pestaña de descargas elegir la versión de nuestro sistema operativo. Una vez se haya descargado el archivo se descomprime y se clican dos veces sobre el ícono señalado en la Figura 29.

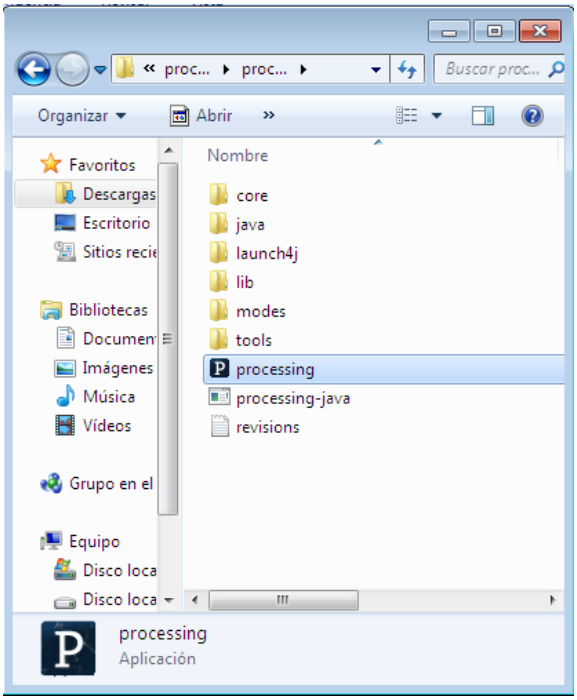


Figura 29: Carpeta de archivos de Processing.

Instalación de librerías

Para poder trabajar con kinect en un PC se necesitan los siguientes elementos:

- OpenNI

Open Natural Interaction, es el software libre facilitado por PrimeSense para comunicarse con Kinect.

- NITE

Es necesario para poder instalar la librería SimpleOpenNI y el que permite el reconocimiento del esqueleto.

- Drivers de los sensores

- Librería SimpleOpenNI

Es un wrapper para poder trabajar con Processing, contiene la mayoría de características de OpenNI pero no todas ya que pretende ser de fácil acceso.

Desde el siguiente enlace: http://code.google.com/p/simple-openni/ se puede descargar una carpeta que contiene todo lo necesario para empezar a trabajar, debe asegurarse de descargar la versión adecuada para el tipo de sistema operativo que se utiliza (Windows, Mac, Linux) y si la versión es de 32 ó 64 bits. En este caso se utiliza Windows 7 de 32 bits.

Primero ejecutamos OpenNI, sólo con esto el ordenador no es capaz de reconocer la kinect como tal sino que aparece como XBOX NUI por lo que lo siguiente es instalar el SensorKinect Driver que venía en la carpeta; el Sensor Win es para otros dispositivos de Prime Sense como AsusXtion Pro.

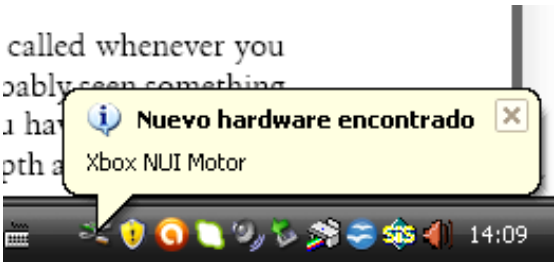


Figura 30: Mensaje de que PC no reconoce kinect.

En este caso el ordenador no es capaz de reconocer los sensores automáticamente por lo que se deben buscar desde el administrador de dispositivos, Figura 31. Clic botón derecho del ratón y se elige actualizar controlador, a continuación se instala de una lista ó ubicación específica. Se examina la carpeta D:\Archivos de programa\PrimeSense\SensorKinect\Driver y se comprueba como ya reconoce el Motor. Se repite la acción para el audio y la cámara.

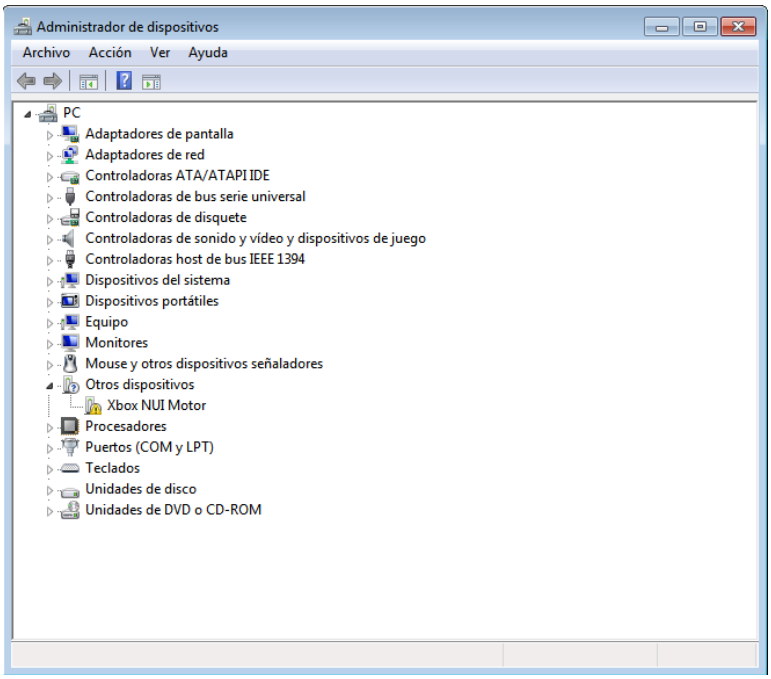


Figura 31: Administrador de dispositivos.

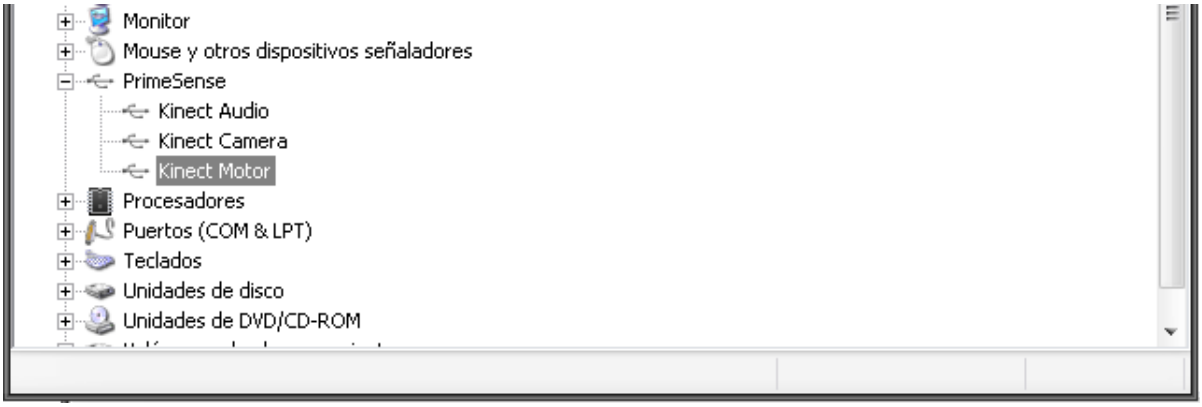


Figura 32: Sensores kinect.

El siguiente paso es instalar la librería SimpleOpenNI para poder trabajar con Processing. Se descarag del mismo link anterior: http://code.google.com/p/simple-openni/downloads/detail?name=SimpleOpenNI-0.27.zip y una vez descomprimida se guarda la carpeta SimpleOpenNI-0.27 dentro de la carpeta *libraries* de Processing, si esta carpeta no existe se tendrá que crear. Un fallo muy común es poner dicha carpeta donde se encuentra *sketch* cuando debe ponerse en la carpeta principal de Processing. Además debe seguirse una jerarquía de carpetas para que Processing reconozca las librerías.

Para este caso: C:\Users\Usuario\Documents\Processing\libraries\SimpleOpenNI. Una vez hecho esto al abrir un *sketch* ya aparecerá SimpleOpenNI seleccionable.

Para utilizar la librería Encog se siguen los mismos pasos. Se descarga la librería de su sitio web https://github.com/encog/encog-java-core/downloads y en la carpeta library se guarda el archivo .JAR[14] .

[14] Java ARchive.

C:\Users\Usuario\Documents\Processing\libraries\encog\library.

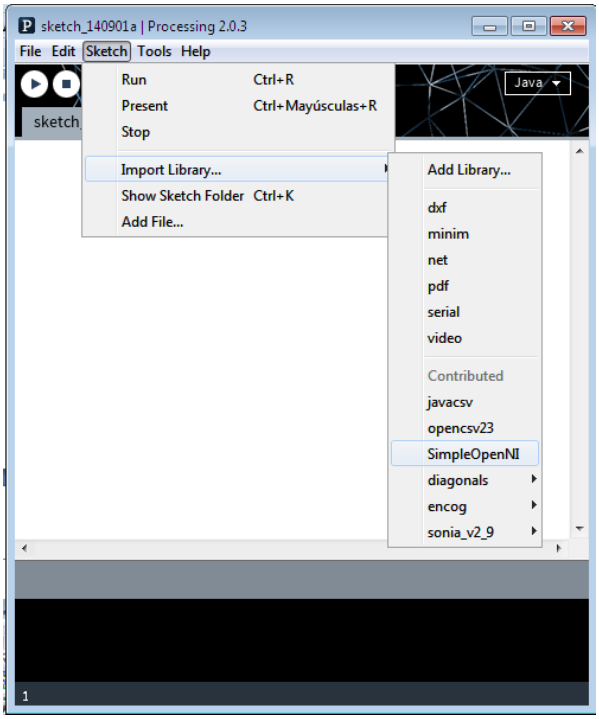


Figura 33: Uso de librerías en Processing.

Conexión de kinect

Para conectar kinect al PC se necesita un cable con una fuente de alimentación externa y un extremo hembra.

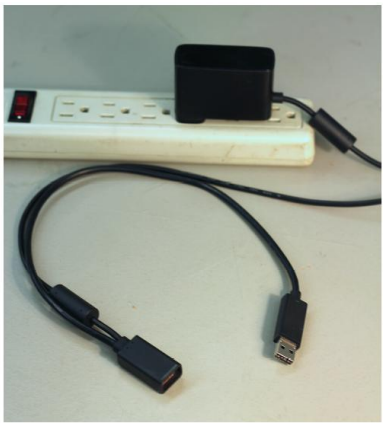


Figura 34: Cable de conexión de kinect.[15]

[15] Greg Borenstein, Making Things See. (2012)

ANEXO B: Resultados de las pruebas para movimientos simples y complejos

Movimientos simples

Prueba 1	1	2	3	4	5	6	7	8
Mov 1	0.125	0.120	0.121	0.129	0.122	0.131	0.129	0.126
Mov 2	0.111	0.131	0.136	0.120	0.123	0.110	0.116	0.139
Mov 3	0.114	0.129	0.133	0.122	0.130	0.115	0.119	0.136
Mov 4	0.121	0.125	0.129	0.127	0.122	0.124	0.125	0.131
Mov 5	0.115	0.127	0.130	0.122	0.122	0.113	0.119	0.137
Mov 6	0.137	0.114	0.113	0.132	0.133	0.149	0.137	0.114
Mov 7	0.136	0.118	0.116	0.135	0.124	0.154	0.139	0.120
Mov 8	0.116	0.128	0.133	0.124	0.123	0.118	0.120	0.135

Tabla 28: Resultado exacto de la primera prueba de movimientos simples.

Prueba 2	1	2	3	4	5	6	7	8
Mov 1	0.148	0.124	0.108	0.118	0.119	0.138	0.160	0.118
Mov 2	0.090	0.141	0.176	0.159	0.150	0.098	0.079	0.163
Mov 3	0.099	0.138	0.164	0.152	0.144	0.103	0.091	0.155
Mov 4	0.108	0.134	0.148	0.142	0.137	0.109	0.102	0.147
Mov 5	0.096	0.136	0.138	0.133	0.129	0.102	0.097	0.144
Mov 6	0.161	0.113	0.096	0.112	0.120	0.154	0.171	0.107
Mov 7	0.179	0.120	0.090	0.108	0.110	0.159	0.207	0.107
Mov 8	0.099	0.137	0.159	0.148	0.142	0.103	0.092	0.153

Tabla 29: Resultado exacto de la segunda prueba de movimientos simples.

Prueba 3	1	2	3	4	5	6	7	8
Mov 1	0.145	0.072	0.089	0.133	0.119	0.154	0.149	0.091
Mov 2	0.056	0.240	0.195	0.095	0.132	0.043	0.078	0.203
Mov 3	0.063	0.211	0.179	0.098	0.130	0.050	0.084	0.186
Mov 4	0.076	0.167	0.155	0.104	0.128	0.065	0.095	0.158
Mov 5	0.086	0.149	0.136	0.109	0.126	0.076	0.102	0.148
Mov 6	0.306	0.025	0.052	0.185	0.124	0.398	0.289	0.045
Mov 7	0.209	0.042	0.062	0.152	0.113	0.248	0.191	0.063
Mov 8	0.063	0.208	0.176	0.098	0.128	0.051	0.083	0.183

Tabla 30: Resultado exacto de la tercera prueba de movimientos simples.

Prueba 4	1	2	3	4	5	6	7	8
Mov 1	0.165	0.050	0.070	0.103	0.118	0.157	0.167	0.063
Mov 2	0.032	0.256	0.220	0.166	0.134	0.019	0.029	0.237
Mov 3	0.039	0.214	0.196	0.157	0.135	0.024	0.035	0.210
Mov 4	0.055	0.159	0.157	0.146	0.133	0.036	0.050	0.165
Mov 5	0.055	0.166	0.157	0.146	0.137	0.035	0.049	0.171
Mov 6	0.368	0.023	0.044	0.078	0.119	0.435	0.376	0.031
Mov 7	0.250	0.032	0.053	0.088	0.115	0.282	0.263	0.041
Mov 8	0.038	0.224	0.198	0.158	0.135	0.023	0.035	0.216

Tabla 31: Resultado exacto de la cuarta prueba de movimientos simples.

Prueba 5	1	2	3	4	5	6	7	8
Mov 1	0.467	1.39E-5	7.1E-4	0.065	0.058	0.003	0.270	0.010
Mov 2	0.002	0.790	0.576	0.004	0.172	1.39E-6	7.28E-4	0.221
Mov 3	0.002	0.423	0.356	0.011	0.159	2.08E-6	0.001	0.249
Mov 4	0.016	0.002	0.023	0.777	0.048	1.22E-5	0.016	0.434
Mov 5	0.017	0.011	0.062	0.149	0.263	1.22E-5	0.007	0.236
Mov 6	0.024	2.09E-7	5.06E-5	0.004	0.004	0.897	0.259	2.99E-4
Mov 7	0.116	6.09E-7	7.94E-5	0.015	0.007	0.193	0.417	0.001
Mov 8	0.002	0.199	0.232	0.041	0.099	2.51E-6	0.002	0.322

Tabla 32: Resultado exacto de la quinta prueba de movimientos simples.

Prueba 6	1	2	3	4	5	6	7	8
Mov 1	0.467	2.88E-5	0.001	0.109	0.034	0.004	0.465	0.002
Mov 2	3.16E-4	0.620	0.522	0.002	0.027	2.29E-5	1.34E-4	0.180
Mov 3	3.38E-5	0.377	0.325	0.013	0.020	3.43E-5	3.03E-4	0.320
Mov 4	0.001	0.007	0.015	0.799	0.008	2.19E-4	0.013	0.698
Mov 5	0.005	0.015	0.063	0.100	0.316	1.03E-4	0.002	0.102
Mov 6	0.022	2.36E-6	1.91E-4	0.019	5.93E-4	0.832	0.286	0.001
Mov 7	0.090	5.93E-6	2.90E-4	0.157	8.66E-4	0.221	0.680	0.006
Mov 8	2.08E-4	0.264	0.169	0.079	0.006	3.80E-5	7.32E-4	0.697

Tabla 33: Resultado exacto de la sexta prueba de movimientos simples.

Prueba 7	1	2	3	4	5	6	7	8
Mov 1	0.621	1.16E-6	0.002	0.096	0.046	0.002	0.380	0.002
Mov 2	4.02E-5	0.850	0.574	0.001	0.056	1.48E-6	5.998	0.183
Mov 3	5.96E-5	0.525	0.398	0.005	0.027	2.07E-6	1.71E-4	0.266
Mov 4	4.09E-4	0.002	0.023	0.708	0.001	7.63E-6	0.013	0.711
Mov 5	0.002	0.021	0.111	0.014	0.194	8.34E-6	0.002	0.033
Mov 6	0.006	4.46E-8	1.89E-4	0.006	8.28E-5	0.899	0.328	0.003
Mov 7	0.020	1.33E-7	3.77E-4	0.089	1.52E-4	0.115	0.546	0.019
Mov 8	2.17E-5	0.255	0.207	0.052	0.002	1.71E-6	5.14E-4	0.777

Tabla 34: Resultado exacto de la séptima prueba de movimientos simples.

Prueba 8	1	2	3	4	5	6	7	8
Mov 1	0.842	7.32E-9	0.001	0.070	0.020	3.69E-4	0.076	3.87E-5
Mov 2	7.21E-6	0.980	0.529	3.96E-5	0.003	4.65E-8	7.52E-6	0.023
Mov 3	2.28E-5	0.401	0.466	4.71E-4	0.011	1.87E-7	2.06E-5	0.028
Mov 4	2.17E-4	6.23E-6	0.006	0.823	1.61E-4	3.92E-5	0.010	0.760
Mov 5	0.009	4.66E-5	0.148	0.007	0.775	9.12E-7	7.49E-5	1.81E-4
Mov 6	0.005	3.5E-10	5.85E-6	0.003	9.47E-6	0.539	0.712	0.001
Mov 7	0.011	5.6E-10	1.65E-5	0.017	1.80E-5	0.091	0.836	0.002
Mov 8	1.208	0.212	0.036	0.003	1.13E-4	1.38E-6	4.49E-4	0.966

Tabla 35: Resultado exacto de la octava prueba de movimientos simples.

Prueba 9	1	2	3	4	5	6	7	8
Mov 1	0.859	2.45E-7	5.99E-4	0.043	0.018	3.36E-4	0.052	4.84E-5
Mov 2	1.14E-5	0.968	0.529	5.11E-5	0.006	1.98E-7	4.61E-6	0.010
Mov 3	3.22E-5	0.456	0.578	2.81E-4	0.028	1.00E-6	1.33E-5	0.014
Mov 4	1.56E-4	8.54E-5	0.004	0.767	0.001	4.61E-5	0.009	0.743
Mov 5	0.004	9.45E-5	0.124	0.004	0.759	6.87E-6	8.02E-5	8.71E-4
Mov 6	0.002	2.26E-7	5.12E-5	0.001	3.31E-5	0.929	0.614	0.005
Mov 7	0.010	2.85E-7	7.42E-5	0.020	3.48E-5	0.092	0.944	0.005
Mov 8	2.94E-6	0.228	0.053	0.007	2.20E-4	1.29E-6	4.21E-4	0.929

Tabla 36: Resultado exacto de la novena prueba de movimientos simples.

Prueba10	1	2	3	4	5	6	7	8
Mov 1	0.862	9.24E-9	0.003	0.009	0.021	7.70E-5	0.054	2.51E-7
Mov 2	2.35E-6	0.992	0.280	3.94E-6	0.003	1.04E-7	7.96E-7	0.006
Mov 3	1.16E-5	0.377	0.542	8.05E-5	0.010	4.10E-7	3.09E-6	0.009
Mov 4	5.88E-6	2.18E-6	0.005	0.766	1.19E-4	3.12E-4	0.003	0.685
Mov 5	0.004	2.71E-5	0.161	0.039	0.597	2.23E-6	4.21E-5	6.31E-5
Mov 6	4.72E-5	1.17E-9	2.35E-6	0.001	9.36E-8	0.984	0.185	5.64E-4
Mov 7	0.006	1.95E-9	1.76E-5	0.033	1.84E-6	0.155	0.843	6.22E-4
Mov 8	3.00E-7	0.088	0.061	0.002	2.59E-5	6.93E-6	1.02E-4	0.879

Tabla 37: Resultado exacto de la décima prueba de movimientos simples.

Movimientos complejos

Prueba 1	1	2	3	4	5	6	7	8
Mov 1	0.269	0.001	0.002	0.274	0.696	0.002	0.002	0.701
Mov 2	5.41E-6	0.986	0.987	3.82E-6	0.019	0.004	0.004	0.019
Mov 3	1.88E-5	0.959	0.961	1.41E-5	0.058	0.002	0.002	0.059
Mov 4	0.022	1.23E-5	1.48E-5	0.022	0.002	0.913	0.926	6.44E-4

Tabla 38: Resultado exacto de la primera prueba de movimientos complejos.

Prueba 2	1	2	3	4	5	6	7	8
Mov 1	0.539	6.58E-4	4.70E-4	0.515	0.439	0.004	0.003	0.450
Mov 2	6.12E-5	0.994	0.993	1.10E-4	0.015	7.07E-4	0.002	0.014
Mov 3	1.62E-4	0.942	0.941	2.79E-4	0.086	3.74E-4	0.001	0.082
Mov 4	0.012	1.46E-4	1.13E-4	0.011	2.97E-4	0.972	0.972	6.51E-4

Tabla 39: Resultado exacto de la segunda prueba de movimientos complejos.

Prueba 3	1	2	3	4	5	6	7	8
Mov 1	0.133	0.001	0.002	0.146	0.861	9.14E-4	9.54E-4	0.867
Mov 2	9.45E-6	0.993	0.993	1.42E-5	0.014	0.002	4.09E-4	0.015
Mov 3	3.18E-5	0.967	0.964	4.41E-5	0.053	0.002	3.68E-4	0.051
Mov 4	0.012	4.91E-4	5.68E-4	0.016	1.75E-4	0.973	0.973	2.57E-4

Tabla 40: Resultado exacto de la tercera prueba de movimientos complejos.

Prueba 4	1	2	3	4	5	6	7	8
Mov 1	0.287	0.013	0.010	0.319	0.245	0.148	0.164	0.202
Mov 2	0.015	0.925	0.909	0.014	0.201	1.80E-4	2.85E-4	0.136
Mov 3	0.053	0.700	0.598	0.054	0.339	0.001	0.002	0.262
Mov4	0.253	0.001	0.002	0.229	0.076	0.721	0.710	0.049

Tabla 41: Resultado exacto de la cuarta prueba de movimientos complejos.

Prueba 5	1	2	3	4	5	6	7	8
Mov 1	0.204	0.005	0.006	0.138	0.033	0.402	0.364	0.026
Mov 2	0.009	0.972	0.979	0.006	0.051	0.001	5.56E-4	0.055
Mov 3	0.018	0.886	0.907	0.010	0.053	0.004	0.002	0.058
Mov 4	0.200	3.99E-4	3.79E-4	0.188	0.015	0.850	0.860	0.013

Tabla 42: Resultado exacto de la quinta prueba de movimientos complejos.

Prueba 6	1	2	3	4	5	6	7	8
Mov 1	0.648	0.017	0.013	0.676	0.119	0.005	0.004	0.120
Mov 2	0.018	0.983	0.977	0.019	0.047	3.34E-5	7.62E-5	0.056
Mov 3	0.034	0.943	0.925	0.039	0.063	6.01E-5	1.14E-4	0.0 67
Mov 4	0.020	9.69E-5	9.26E-5	0.025	0.147	0.871	0.840	0.128

Tabla 43: Resultado exacto de la sexta prueba de movimientos complejos.

Prueba 7	1	2	3	4	5	6	7	8
Mov 1	0.002	0.138	0.118	0.002	0.197	0.006	0.004	0.203
Mov 2	1.65E-4	0.991	0.985	2.23E-4	0.070	2.83E-4	8.67E-4	0.066
Mov 3	1.76E-4	0.980	0.971	2.43E-4	0.076	3.67E-4	1.22E-4	0.067
Mov 4	0.004	0.003	0.002	0.004	0.106	0.488	0.525	0.132

Tabla 44: Resultado exacto de la séptima prueba de movimientos complejos.

Prueba 8	1	2	3	4	5	6	7	8
Mov 1	0.275	0.012	0.010	0.360	0.142	0.013	0.015	0.100
Mov 2	0.002	0.993	0.993	9.86E-4	0.017	3.27E-6	2.04E-5	0.016
Mov 3	0.003	0.959	0.955	0.003	0.040	2.58E-5	1.10E-4	0.031
Mov 4	0.003	1.60E-4	1.90E-4	0.005	0.540	0.772	0.792	0.521

Tabla 45: Resultado exacto de la octava prueba de movimientos complejos.

Prueba 9	1	2	3	4	5	6	7	8
Mov 1	0.204	0.155	0.261	0.164	6.27E-5	0.003	0.002	1.00E-4
Mov 2	0.001	0.996	0.993	4.63E-4	9.23E-4	2.89E-5	8.29E-6	0.001
Mov 3	0.002	0.992	0.988	0.001	6.27E-4	4.09E-5	1.27E-5	7.41E-4
Mov 4	0.001	9.62E-4	0.001	8.11E-4	0.050	0.921	0.886	0.046

Tabla 46: Resultado exacto de la novena prueba de movimientos complejos.

Prueba10	1	2	3	4	5	6	7	8
Mov 1	0.936	0.070	0.052	0.942	0.001	1.45E-4	1.29E-4	0.002
Mov 2	0.014	0.997	0.998	0.017	0.001	2.36E-7	6.22E-8	0.001
Mov 3	0.038	0.991	0.991	0.049	0.001	4.35E-7	1.74E-7	0.002
Mov 4	0.001	4.75E-5	4.17E-5	0.002	0.484	0.740	0.768	0.507

Tabla 47: Resultado exacto de la décima prueba de movimientos complejos.